Fusion of Phoneme Recognisers for South African English

George Wessel Strydom

Thesis presented in partial fulfilment of the requirements for the degree
Master of Science in Electronic Engineering
at the University of Stellenbosch

Supervisor: Prof J.A. du Preez

December 2008

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

SIGNATURE DATE

Abstract

Phoneme recognition systems typically suffer from low classification accuracy. Recognition for South African English is especially difficult, due to the variety of vastly different accent groups. This thesis investigates whether a fusion of classifiers, each trained on a specific accent group, can outperform a single general classifier trained on all.

We implemented basic voting and score fusion techniques from which a small increase in classifier accuracy could be seen. To ensure that similarly-valued output scores from different classifiers imply the same opinion, these classifiers need to be calibrated before fusion. The main focus point of this thesis is calibration with the Pool Adjacent Violators algorithm. We achieved impressive gains in accuracy with this method and an in-depth investigation was made into the role of the prior and the connection with the proportion of target to non-target scores.

Calibration and fusion using the information metric C_{llr} was showed to perform impressively with synthetic data, but minor increases in accuracy was found for our phoneme recognition system. The best results for this technique was achieved by calibrating each classifier individually, fusing these calibrated classifiers and then finally calibrating the fused system.

Boosting and Bagging classifiers were also briefly investigated as possible phoneme recognisers. Our attempt did not achieve the target accuracy of the classifier trained on all the accent groups.

The inherent difficulties typical of phoneme recognition were highlighted. Low per-class accuracies, a large number of classes and an unbalanced speech corpus all had a negative influence on the effectivity of the tested calibration and fusion techniques.

Opsomming

Foneemherkenningstelsels het tipies lae klassifikasie akkuraatheid. As gevolg van die verskeidenheid verskillende aksent groepe is herkenning vir Suid-Afrikaanse Engels veral moeilik. Hierdie tesis ondersoek of 'n fusie van klassifiseerders, elk afgerig op 'n spesifieke aksent groep, beter kan doen as 'n enkele klassifiseerder wat op alle groepe afgerig is.

Ons het basiese stem- en tellingfusie tegnieke geïmplementeer, wat tot 'n klein verbetering in klassifiseerder akkuraatheid gelei het. Om te verseker dat soortgelyke uittreetellings van verskillende klassifiseerders dieselfde opinie impliseer, moet hierdie klassifiseerders gekalibreer word voor fusie. Die hoof fokuspunt van hierdie tesis is kalibrasie met die *Pool Adjacent Violators* algoritme. Indrukwekkende toenames in akkuraatheid is behaal met hierdie metode en 'n in-diepte ondersoek is ingestel oor die rol van die aanneemlikheidswaarskynlikhede en die verwantskap met die verhouding van teiken tot nie-teiken tellings.

Kalibrasie en fusie met behulp van die informasie maatstaf C_{llr} lewer indrukwekkende resultate met sintetiese data, maar slegs klein verbeterings in akkuraatheid is gevind vir ons foneemherkenningstelsel. Die beste resultate vir hierdie tegniek is verkry deur elke klassifiseerder afsonderlik te kalibreer, hierdie gekalibreerde klassifiseerders dan te kombineer en dan die finale gekombineerde stelsel weer te kalibreer.

Boosting en *Bagging* klassifiseerders is ook kortliks ondersoek as moontlike foneem herkenners. Ons poging het nie die akkuraatheid van ons basislyn klassifiseerder (wat op alle data afgerig is) bereik nie.

Die inherente probleme wat tipies is tot foneemherkenning is uitgewys. Lae per-klas akkuraatheid, 'n groot hoeveelheid klasse en 'n ongebalanseerde spraak korpus het almal 'n negatiewe invloed op die effektiwiteit van die getoetsde kalibrasie en fusie tegnieke gehad.

Acknowledgements

I would like to thank the following people, without whom this thesis would never have been possible:

- Prof. J.A. du Preez, for his guidance, ideas and motivational speeches during the difficult times.
- My family, for their support, especially financially.
- My friends, for supplying me with distractions when I needed it most.
- Everyone in the DSP lab, for all the help, lunches, coffees, friendships and a great work environment.
- Everyone at Blue Cube Systems, for the support and allowing me the opportunity to finish this thesis.
- Niko Brümmer, for his work in calibration and fusion and the FoCal toolkits.
- Everyone who has made an addition to the PATRECII library.
- And lastly, Esther van Heerden, for her love, support, and putting up with me during this past year.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

AST	African Speech Technology
DSP	Digital Signal Processing
DFT	Discrete Fourier Transform
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
LDA	Linear Discriminant Analysis
LRE	Language Recognition Evaluation
MFCC	Mel-scale Frequency Ceptral Components
PAV	Pool Adjacent Violators
PCA	Principal Component Analysis
PDF	Probability Density Function
SAMME	Stagewise Additive Modeling using a Multi-class Exponential loss function
SRE	Speaker Recognition Evaluation
SVM	Support Vector Machine
T-BAGMM	Tree-Based Adaptive Gaussian Mixture Models

Variables

symbol	**description**
f	Frequency
σ	Standard deviation
μ	Mean
x	Feature
y	Class label
w_i	Weight
g_i	Tree node weight
K	Number of components or systems
S_i	State
a_{ij}	Transition probability

symbol	**description**
$\vec{s}$	Vector of scores
N	Number of classes
H_i	Class
$\vec{\ell}$	Vector of log-likelihoods
P	Prior probability
T	Number of trials
t	Trial
α	Scaling constant
$\vec{\beta}$	Offset vector
$\mathcal{L}$	Dataset
err	Error

Functions

$\mathcal{T}(x)$	The classifier accepting input feature x.
$\mathbb{I}(\ldots)$	Logical operator (outputs one if true, zero if false)
$p(x)$	The probability density function with respect to variable x
$P(x)$	The probability of random variable X having the value x
$P(x\|H)$	The conditional probability of the variable x given the knowledge H.

Chapter 1

Introduction

1.1 Motivation

English in South Africa is spoken by a large variety of vastly different accent groups. This difference has a negative influence on the accuracy of automatic speech recognition systems.

The African Speech Technology (AST) Project [23] collected and transcribed speech data for various South African languages. In the specific case of English, data for five accent groups were recorded. These subsets can be used to train a generalised classifier that is fairly accurate for all accents, or five separate classifiers, each an expert for one specific accent. The question that this thesis will try to answer is whether a fusion of these expert systems will be more accurate than the general classifier.

The concept of fusing a multitude of independent opinions is nothing new. Democratic governments are elected by a majority vote. Online stores use a fusion of review scores to inform the customer of the relative quality of a product. The larger the voting base, the more accurate we expect the resulting fusion to be.

In the last few years fusion has been successfully used in the NIST LRE (Language Recognition Evaluations) [19] and the NIST SRE (Speaker Recognition Evaluations) [20]. These systems are all based on speech, with classifiers based on different metrics and/or models. Fusions of various Gaussian Mixture Models (GMMs) and Support Vector Machines (SVMs) are typical.

While phoneme recognition systems closely resemble speaker or language recognisers, we believe a closer examination is still necessary. The statistical models and speech techniques might be the same, but, in general, phoneme recognisers are much less accurate and the classification problems usually contain more classes to be recognised. This inaccuracy and high number of classes might make some fusion techniques unsuitable for phoneme recognition.

1.2 Background

In the context of this thesis, the word *fusion* refers to the combination of a number of individual system *decisions* to form a single *decision*. These decisions can be hard or soft.

Hard decisions refer to a definite answer, for example the output of a specific class label from a classifier. Soft decisions refer to the output of a real number or a vector of real numbers, also referred to as *scores*. It can be turned into hard decisions, typically by thresholding or selecting the maximum score.

A score can be any real number that has meaning in the specific context of that system. Probabilities, likelihoods and likelihood-ratios are all examples of commonly used scores. In the case of a classifier, the maximum score usually indicates the most likely class and the minimum score the least likely.

Calibration ensures that the scores from each classifier have the same *meaning*, for example that an output score from one classifier implies the same as a similarly-valued output score from another. The output scores of a classifier can have the potential for acceptable accuracy, but suffer from a calibration problem, thus resulting in lower accuracies. Calibration of these scores before classification helps prevent such problems. The fusion of scores from different systems can also be improved by a calibration of each system beforehand.

Phonemes are a chosen set of speech sounds, that when used together, form the spoken words of a language [16]. These phoneme sets usually differ, depending on the particular language that is modelled. One all-encompassing set therefore does not exist. The phoneme set for a language is typically much larger than the alphabet for that specific language. The '*a*' in *cat* and *cast*, for example, does not have the same pronunciation.

A *phoneme recogniser* accepts speech feature vectors as input and outputs phoneme labels (or vectors of scores from which a decision about these labels can be made). Hidden Markov Models (HMMs) are currently the most widely used statistical model for this purpose [22].

1.3 Literature synopsis

The concepts of fusion and calibration are not new in the field of pattern recognition, but in recent years certain techniques have received much attention. These have greatly improved the accuracies of language and speaker recognition systems and have been very successfully applied in the NIST LRE (Language Recognition Evaluation) [19] and SRE (Speaker Recognition Evaluation) [20].

Voting methods are a popular method of combining the decisions from classifiers. A comparison of voting methods by Van Erp [13] showed that plurality voting is a simple and fast method of classifier combination. Other methods were also investigated, with some achieving better results than plurality voting.

The Pool Adjacent Violators (PAV) Algorithm was first published in [1] and has recently begun to be used for the purpose of calibrating the output scores of pattern recognition systems. This algorithm finds a stepwise monotonic function that maps scores to posterior probabilities [25] from which the log-likelihood-ratios can be recovered [6]. The optimality of the PAV algorithm for binary proper scoring rules is proved in [7]. Proper scoring rules are special cost functions to judge the cost-effective decision-making ability of predictions in

the form of posterior probabilities.

Brümmer[5][6][8] proposed an information-theoretic measure C_{llr}. This metric gives a measure of the total quality of the information delivered to the user (after calibration loss) by a classifier. As opposed to posterior probabilities, C_{llr} measures the goodness of log-likelihood-ratios. Optimising the objective function C_{llr} results in a metric of the loss caused by poor calibration. This information lost by poor calibration can be retrieved with a calibration transformation.

A linear transformation with scaling and translation can be trained by optimising C_{llr} using logistic regression [5][6][8][24]. The logistic regression model and a comparison of numerical optimisers for logistic regression can be found in [18].

Bagging (**B**ootstrap **Agg**regat**ing**) [2] is a technique where instead of calibrating and fusing existing, previously trained classifiers, it involves the training of multiple versions of a classifier and the aggregation thereof. Each classifier is trained with a subset of the original database, drawn randomly and with replacement. The predictions of the classifiers are combined with plurality voting into a single prediction.

Boosting is related to Bagging in the sense that a multitude of weak classifiers are trained from a subset of a database. Freund and Schapire [14] introduced Boosting with their AdaBoost algorithm. In the AdaBoost algorithm the classifier at each iteration is trained from a weighted dataset, with weights calculated according to the incorrect classifications of the previous iteration. The weight of each incorrectly classified data point is increased in an attempt to improve the classification of that specific feature in the next iteration. AdaBoost was designed with a two-class problem in mind and can easily become unstable in a situation with a large number of classes. A multi-class AdaBoost algorithm (SAMME) was proposed by [26] that reduced this restriction.

When work was first started on this thesis, no literature on fusion specifically related to phoneme recognition could be found.

1.4 Objectives of this study

- To implement and test various fusion and calibration techniques on the different accent group subsets of the AST South African English dataset.
- To comment on the feasibility of fusing these expert systems, compared to a general model.
- To implement and test the Boosting and Bagging classifiers as possible phoneme recognisers.

1.5 Contributions

- Added various classes to the PATRECII system (pattern recognition system used and developed by the Digital Signal Processing (DSP) group of Stellenbosch University) to make fusion and calibration possible.
- Implemented fusion and calibration algorithms.
- Tested basic fusion techniques such as a voting system and basic score fusion and showed an increase in classifier accuracy.
- Experimented with calibration using the Pool Adjacent Violators Algorithm. Using synthetic data, an in-depth investigation was done into the role of the prior and the connection with the proportion of the target to non-target scores used for PAV training. Impressive gains for a fusion after PAV calibration was shown for our phoneme recognisers.
- Experimented with fusion and calibration using the metric C_{llr} and Linear Logistic Regression. Impressive accuracies was shown with synthetic data, but a minor increase in classification accuracy was achieved for our phoneme recognition system.
- Implemented and tested Bagging and Boosting for comparison purposes. We did not achieve the target accuracy of the classifier trained on the accent groups.
- Highlighted the difficulty when fusing or calibrating phoneme recognisers. The large number of classes, low classifier accuracy and the the unbalanced nature of the typical speech corpus makes calibrating and fusing phoneme recognisers non-trivial.

1.6 Overview of this work

This section gives a synopsis of the work. A brief description of each chapter and the relevance thereof in context of the whole will be shown.

1.6.1 Theory

Chapter 2 (page 14) deals with information needed to understand the inner workings of a phoneme recogniser. This is not an in-depth discussion, but covers the basic concepts of the systems we used. We describe the signal processing phase, from pre-processing to the final feature extraction and normalisation. Probability Density Functions (PDFs), Mixture Models and Hidden Markov Models (HMMs) are briefly explained. We end this section with a description of a basic phoneme model.

Fusion from hard decisions is covered in Chapter 3 (page 21). This involves combining the decisions made by classifiers into one final decision. First a basic plurality voting system

is explained, where each system gets an equal vote. From there we move on to a weighted voting system, where the decisions of systems are not weighted equally. We also describe a system where a constructed score is generated according to the phoneme class and place of articulation. Instead of fusing the original decisions, these constructed scores are fused. The hope is that a system that distinguishes between correct, wrong and almost correct decisions will be more accurate.

Chapter 4 introduces the concept of calibration and the Pool Adjacent Violators algorithm. A brief PAV example is shown on page 26. The PAV algorithm is used to find a mapping from uncalibrated scores to calibrated posterior probabilities. Classifier scores are divided into target and non-target scores. A target score is defined as the output score from a class model to which the input feature belongs. In contrast, a non-target score is the output score from a class model to which the input feature does not belong. The algorithm itself is very basic and the training of a PAV curve can be summarised in four steps:

1. Choose the target and non-target scores.
2. Assign a posterior probability of one to all target scores and zero to all non-target scores.
3. Sort the vector of ones and zeroes according to the values of the target and non-target scores.
4. Iteratively pool all adjacent values that violates monotonicity and replace them with the pool mean.

Synthetic data is used to investigate the role of the prior and its relation to the proportion of target to non-target scores. For a calibration where all classes are to be treated as equal (flat prior), an equal number of target and non-target scores are chosen. To take the prior probabilities into account, the target and non-target scores have to be chosen in the proportion of this prior probability.

PAV will later be used to calibrate the output scores of each class of each classifier before fusion.

Chapter 5 (page 39) introduces fusion of the output scores of various classifiers. Where in Section 3 a decision is made before fusion, here no hard decision is made until after the scores are fused. Once again each system can be weighted equally, or if knowledge exists about the individual system accuracy, the more accurate systems can be given a larger weight. The weighted fusion of scores is defined as

$$\vec{s} = \sum_{k=1}^{K} w_k \vec{s_k} \tag{1.1}$$

with scores $\vec{s}$, weights w and number of classifiers K. For the equally weighted system $w_k = \frac{1}{K}$.

A metric to measure the information delivered by a classifier and the use of that metric for calibration and fusion is introduced in Section 5.3 (page 41). This metric is known as C_{llr} and is defined as

$$C_{llr} = -\frac{1}{T}\sum_{t=1}^{T} w_t log_2 P_t \tag{1.2}$$

where P_t is the posterior probability $P(H_t|x_t)$ of the true class of trial t. The weight w_t is necessary to normalise the class proportions in the evaluation trials.

By optimising for a minimum C_{llr} we can train a linear transformation that can be used for calibration,

$$\vec{\ell}'(x_t) = \alpha\vec{\ell}(x_t) + \vec{\beta}, \tag{1.3}$$

or calibration and fusion in one step,

$$\vec{\ell}'(x_t) = \sum_{k=1}^{K} \alpha_k\vec{\ell}_k(x_t) + \vec{\beta}, \tag{1.4}$$

with K recognisers, scaling constants α_k, offset vector $\vec{\beta}$ and input log-likelihood score vector $\vec{\ell}_k(x_t)$ for input feature x of trial t.

In Chapter 6 (page 46) the Bagging and Boosting algorithms are explained. They are not fusion techniques in the sense that they can be used to fuse existing systems, but are relevant due to the role of fusion in the algorithms themselves. Both Bagging and Boosting train a multitude of weak classifiers from subsets of the training data. The subsets for Bagging is randomly drawn with replacement. The subsets for Boosting uses all the data, but weighs the vectors according to the accuracy of the classifier trained in the previous iteration. AdaBoost is a well known algorithm for Boosting. In the multi-class case AdaBoost can easily become unstable for classifiers with large classification error. An extension of AdaBoost, called SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function), relieves this restriction and stays stable even with much larger classification error.

1.6.2 Experimental Investigation

With all the theory covered in the previous chapters, Chapter 7 (page 50) contains the description of our test systems and the results from the experiments. This chapter starts by giving more information on the African Speech Technology (AST) Corpus for South African English (page 50) and the various accent groups it contains. The five accent groups are referred to as `AE`, `BE`, `CE`, `EE` and `IE`.

In Section 7.3 (page 51) we detail the basic topology of our fusion system. We describe the models used for the classifier for each of the five accent groups. The 44 phonemes from our chosen phoneme set are referred to, as well as the 137 phoneme classes resulting

from most phonemes consisting of multiple states. These multiple states exist to accurately model phonemes for which the vocal tract does not stay constant for the total duration of the phoneme.

Due to low classifier accuracy, high memory requirements and the large amount of training data needed, we reduce the 137 classes to 44, one for each phoneme. The scores from the 137-dimensional output log-likelihood vector of each classifier can not be combined as is, but need to be converted to posterior probabilities. These posterior probabilities for all the states of a phoneme can then be added to form a single score. Using Bayes' Theorem for this mapping and the choice of priors are explained in Section 7.4 (page 53). We decided to not choose priors according to phoneme occurrence in the dataset, but rather give an equal prior to each phoneme. For each phoneme, this prior was then split amongst its states.

Individual classifiers

The accuracy of the individual classifiers before any calibration or fusion is shown in Section 7.5 (page 54). Table 1.1 clearly shows that on all the data, the `ALL` system (trained on all accent groups) is superior. For the 137-class case we also compared classification from log-likelihoods with that from posterior probabilities. The mapping to posteriors with our chosen prior had a positive effect on classification accuracy, possibly due to the high accuracy and frequent occurrence of the {`sil`} phoneme. This specific phoneme consists of only one state, thus receiving a larger prior. The larger prior results in {`sil`} slightly dominating the lesser frequent phonemes, increasing the total classifier accuracy.

	AE	BE	CE	EE	IE	ALL
137 classes - log-likelihoods	39.916%	33.648%	38.721%	40.490%	37.127%	47.748%
137 classes - posteriors	41.509%	35.510%	40.311%	41.937%	39.252%	49.443%
44 classes - log-likelihoods	45.588%	39.784%	44.657%	45.828%	43.350%	50.296%

Table 1.1: *The accuracies of individual classifiers. Note the relatively low accuracy of the `BE` classifiers when compared to that of the other accent groups. Our chosen prior has an advantageous effect on the accuracy of the most frequent class* {`sil`} *and results in a slight gain in total accuracy.*

Voting systems

From Section 7.6 (page 55) we attempt out first fusion experiments with the use of basic voting systems. In Tables 1.2 and 1.3 we can see that even a basic method such as voting already shows some promise, with the combination of the five accent groups systems achieving a higher accuracy than the `ALL` system. Usage of the phoneme characteristics (the phoneme class and place of articulation) does not achieve the same level of accuracy as the basic voting system.

Equally weighted voting system (137 classes)	48.528%
Voting system, weighted according to classifier accuracy (137 classes)	46.874%
Voting with phoneme classes and place of articulation (137 classes)	45.944%

Table 1.2: *The accuracies for hard-decision fusion systems (137 classes). The equally weighted voting system achieves a small increase in accuracy when compared to the 47.748% of our baseline* `ALL` *classifier.*

Equally weighted voting system (44 classes)	53.583%
Voting system, weighted according to classifier accuracy (44 classes)	52.096%
Voting with phoneme classes and place of articulation (44 classes)	51.684%

Table 1.3: *The accuracies for hard-decision fusion systems (44 classes). All these systems achieve an increase in accuracy when compared to the 50.296% of our baseline* `ALL` *classifier.*

Basic score fusion

Basic score fusion in Section 7.7 (page 56) also shows a significant increase in accuracy from the original systems (Tables 1.4 and 1.5). An interesting observation to note is that weighting each system according to its accuracy (the higher the system accuracy, the larger the weight) does not show a noticeable difference from the equally weighted system.

Equally weighted score fusion	45.429%
Weighted score fusion	45.453%

Table 1.4: *The accuracies for score fusion system (137 classes).*

Equally weighted score fusion	53.979%
Weighted score fusion (44 classes)	53.946%

Table 1.5: *The accuracies for score fusion system (44 classes).*

Calibration with the Pool Adjacent Violators algorithm

Our first calibration experiment starts in Section 7.8 (page 57) with the Pool Adjacent Violators algorithm. The PAV algorithm is typically used for binary classification, but we attempted its application for our specific multi-class case. We trained a PAV curve for each class of each classifier and thus needed to divide scores into target and non-target groups. Experiments were done for both the flat prior (equal number of target and non-target scores) and the prior according to phoneme occurrence.

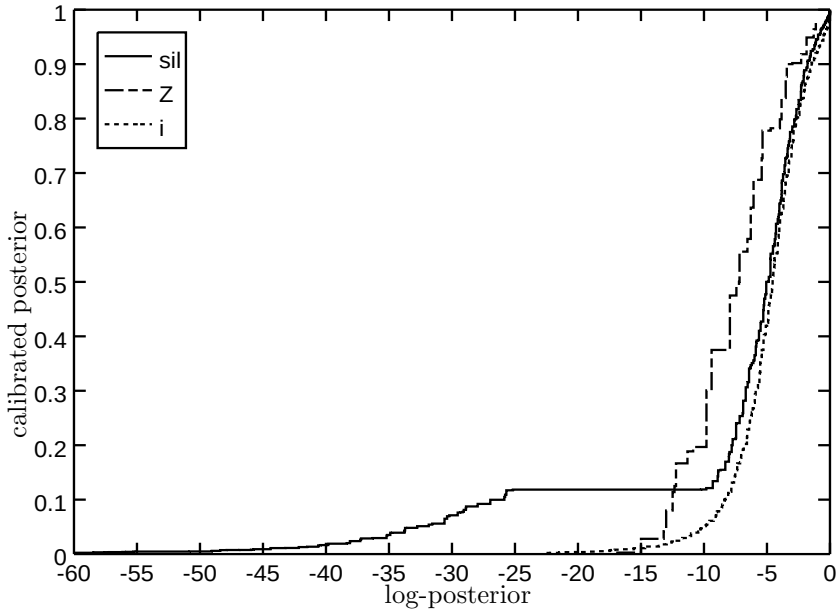


Figure 1.1: *The 44 class trained PAV curves with an equal number of target and non-target scores for the phonemes* {`sil`}, {`Z`} *and* {`i`}. *These phonemes represent an example of the most frequent phoneme, one of the least frequent and a typically occurring phoneme.*

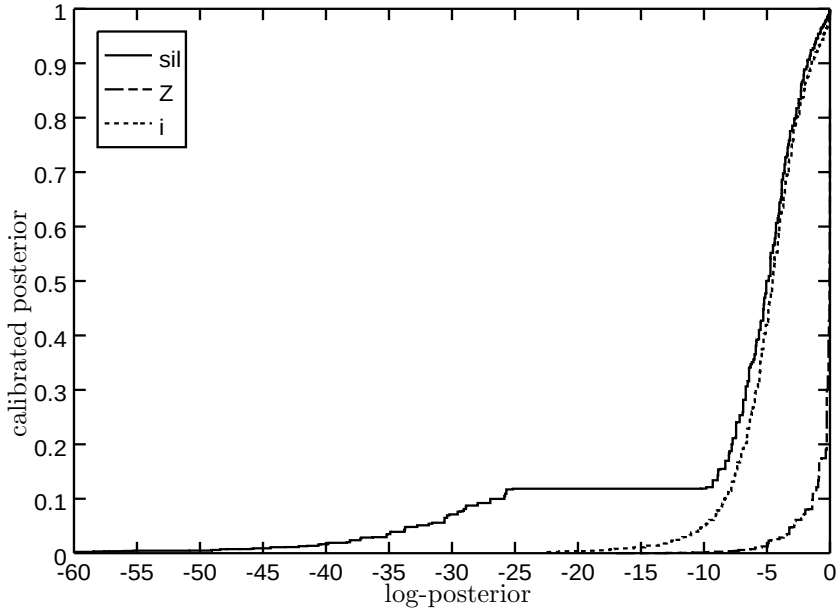


Figure 1.2: *The 44 class trained PAV curves with target and non-target scores in proportion of the class occurrence for the phonemes* {`sil`}, {`Z`} *and* {`i`}. *These phonemes represent an example of the most frequent phoneme, one of the least frequent and a typically occurring phoneme.*

Figures 1.1 and 1.2 show the trained PAV curves for the flat prior and actual prior case. The effect of the prior can clearly be seen in the latter with the log-likelihood scores of the scarce phoneme {Z} mapped to lower posterior probabilities than the other two more frequent phonemes. The stair shape of the {Z} curve, causes by a scarceness of training data, is also easily observable.

The accuracies for the individual classifiers after PAV calibration can be seen in Table 1.6. A PAV calibration with a flat prior has almost no effect on the accuracy of the 44-class classifiers, but a large decrease in accuracy can be seen with 137-class classification. A possible explanation for this is the large number of classes and the resulting scarceness of training data for each class. In both the 44- and 137-class classifiers, the accuracy for some classes increase, while a decrease is shown for others.

If the proportion of target to non-target scores are chosen according to phoneme occurrence, the accuracies show impressive gains. The least frequent classes are suppressed in favour of the more frequent ones, resulting in large gains to the total classifier accuracies.

	AE	BE	CE	EE	IE	ALL
44-class, without PAV	45.59%	39.78%	44.66%	45.83%	43.35%	50.30%
44-class, flat prior	45.04%	39.54%	44.60%	46.26%	43.69%	50.11%
44-class, actual prior	52.46%	47.26%	52.27%	53.18%	51.23%	57.59%
137-class, without PAV	39.92%	33.65%	38.72%	40.49%	37.13%	47.75%
137-class, flat prior	28.00%	24.40%	26.12%	27.50%	26.10%	30.06%
137-lass, actual prior	42.72%	37.22%	42.07%	43.17%	40.77%	47.42%

Table 1.6: *The effect of PAV calibration on system accuracy. PAV calibration, with the number of target and non-target scores in proportion to class frequency, shows impressive results in the 44-class case.*

Table 1.7 list the accuracies of fused systems after PAV calibration. The calibrated posterior probabilities of the five classifiers AE, BE, CE, EE and IE are fused by basic score averaging. The 44-class system, with an equal number of target and non-target scores, shows a slight increase when compared to the baseline ALL classifier. When choosing the target and non-target scores according to the class occurrence, a large gain in accuracy is observed. Both of the 137-class fusions failed to achieve the target accuracy of the baseline classifier.

Calibration and Fusion using the information metric C_{llr}

Section 7.9 covers our experiments with the metric C_{llr}. We used the FoCal Multi-class[4] MATLAB toolkit to train the parameters for the linear transformation used in the fusion and calibration. The unoptimised value of C_{llr} represents the total information loss, in units of bits of information. Optimisation leads to a minimum C_{llr} value. The difference between these two values is the calibration loss, also defined as the loss that can be recovered

44-class, ALL classifier	50.30%
44-class fused, PAV with flat prior	52.70%
44-class fused, PAV with actual prior	59.40%
137-class, ALL classifier	47.75%
137-class fused, PAV with flat prior	34.10%
137-class fused, PAV with actual prior	46.20%

Table 1.7: *The accuracies for the equally weighted score fused systems after PAV calibration. The fusion after a PAV calibration, with the target and non-target scores in proportion of the phoneme occurrence, shows impressive results in the 44-class case.*

through calibration. The refinement loss is the unrecoverable information loss, denoted by the optimised C_{llr}.

The calibration and refinement losses for each classifier are calculated and shown in Figure 1.3. The reference C_{llr} value of a useless, but perfectly calibrated classifier is shown as the first bar in the graph. As expected from the classifier accuracy, the `BE` classifier shows the highest loss.

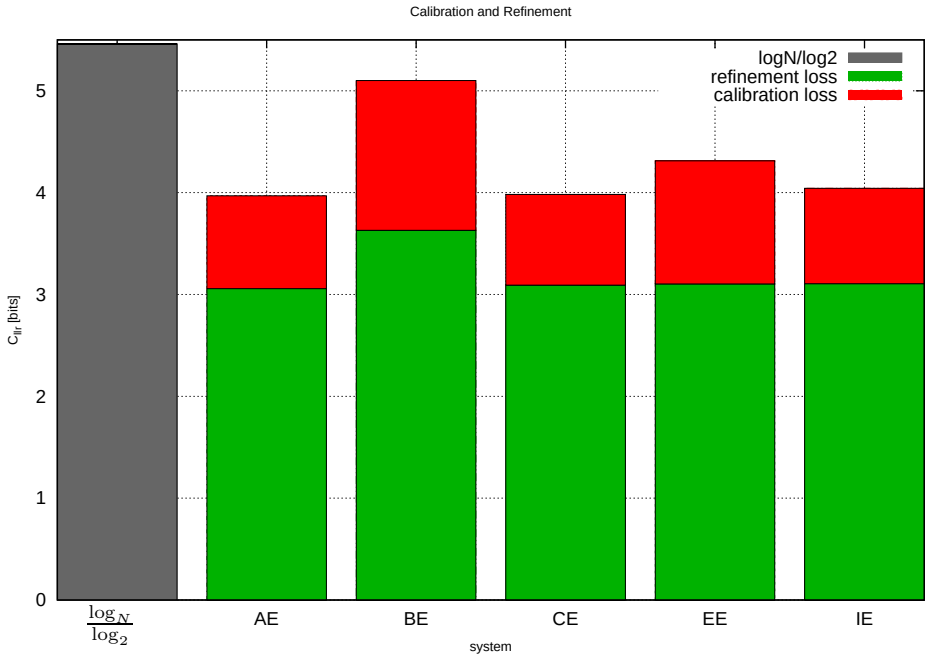


Figure 1.3: *The calibration and refinement loss of each classifier, with the `BE` classifier showing the highest loss.*

We then trained the parameters needed for the calibration and fusion. The trained weight for each classifier is shown in Table 1.8. In comparison to the weights of the other classifiers, the `BE` classifier's weight is very small, at less than half of the second smallest weight. This small weight might have a negative influence on the total accuracy of the fused system, with

results shown in Table 1.9. With such a small weight, the input of the BE system is practically ignored and the accuracy on the BE data should be lower. Another fusion was attempted with each classifier calibrated individually (according to Equation 1.3). This resulted in the marginally better results also listed in Table 1.9. This fused system still contains some calibration loss that can be recovered with an additional calibration. The results for this final fused and calibrated classifier can also be seen in the above mentioned table. A slight increase in accuracy is shown if compared to our reference ALL classifier, with a classification accuracy of 50.296%.

AE	BE	CE	EE	IE
0.149	0.055	0.124	0.128	0.166

Table 1.8: *The FoCal trained weight for classifiers AE, BE, CE, EE and IE (accurate to three decimal places).*

FoCal calibration and fusion	49.910%
FoCal individual calibration and fusion	51.152%
Additional FoCal calibration and fusion	51.235%

Table 1.9: *The accuracies for FoCal calibration and fusion (44 class). The individual calibration and fusion shows a slight increase in accuracy when compared to our baseline ALL classifier with an accuracy of 50.296%.*

Bagging and Boosting

We experimented with the feasibility of the Bagging and Boosting algorithms for phoneme recognition in Section 7.10 (page 70). For these experiments we chose to investigate nine situations with the number of fused classifiers at 64, 128 and 256. Each of these were used in combination with three classifier types, namely a single Diagonal Gaussian Model, a GMM with eight components and a GMM with 16 components.

The results for Bagging is shown in Table 1.10. None of the attempts reached the target accuracy of the ALL classifier. We also expected results to increase according to the total number of classifiers trained and fused, but unfortunately this was not the case.

The Boosting classification results in Table 1.11 show the same trend as the results for the Bagging experiments. The number of classifiers trained had no major effect on the classification results. Our choice of weak classifiers might suffer from accuracies too low for the Bagging or Boosting algorithms to be of much use, but more complex models would have been unpractical. Training times for Bagging and Boosting with these specific weak classifiers already exceeded the training time of our individual classifiers.

	64 classifiers	128 classifiers	256 classifiers
137-class Diagonal Gaussian	36.721%	36.719%	36.718%
44-class Diagonal Gaussian	40.685%	40.683%	40.679%
137-class 8-component GMM	42.378%	42.358%	42.352%
44-class 8-component GMM	46.325%	46.302%	46.325%
137-class 16-component GMM	43.857%	43.852%	43.796%
44-class 16-component GMM	47.731%	47.714%	47.667%

Table 1.10: *The resulting accuracies of Bagging classifiers for several internal classifier types (1, 8 and 16 component Gaussian Mixture Models) and total number of classifiers (64, 128 and 256) for both 137 and 44 classes. The accuracy of our baseline* `ALL` *classifier is 47.75% (137 class) and 50.296% (44 class).*

The fact that we expected a Boosting classifier to achieve better results than a Bagging classifier also invokes some suspicion, but due to time constraints further investigation could not be attempted.

	64 iterations	128 iterations	256 iterations
137-class Diagonal Gaussian	36.380%	36.892%	36.780%
44-class Diagonal Gaussian	39.643%	40.138%	39.937%
137-class 8-component GMM	42.708%	41.677%	42.812%
44-class 8-component GMM	45.020%	44.418%	45.138%
137-class 16-component GMM	43.001%	44.093%	43.921%
44-class 16-component GMM	45.500%	46.256%	45.876%

Table 1.11: *The resulting accuracies of Boosting classifiers for several internal classifier types (1, 8 and 16 component Gaussian Mixture Models) and total number of iterations (64, 128 and 256) for both 137 and 44 classes. The accuracy of our baseline* `ALL` *classifier is 47.75% (137 class) and 50.296% (44 class).*

Chapter 2

Phoneme Recognition Basics

2.1 Introduction

In this chapter we cover the basics needed to understand the specific models and techniques used to implement a phoneme recogniser. We do not include various alternative methods, but only focus on those used in our own systems. A block diagram of the basic topology of a phoneme recognition system can be seen in Figure 2.1.

Section 2.2 provides a description of the signal processing phase, where raw speech data is transformed into feature vectors. The basics of the Probability Density Function (PDF) is covered in Section 2.3. Section 2.4 expands on the PDF by introducing Mixture Models. An introduction to the basic theory behind Hidden Markov Models (HMMs) is given in Section 2.5. Section 2.6 shows how the models in the previous sections are combined to form a phoneme model.

2.2 Signal Processing

In this section we briefly introduce the basics needed to understand the signal processing phase of a phoneme recognition system. Once again we only focus on the techniques applicable to our system. Figure 2.2 shows a block diagram with detail on the individual steps that turns raw speech data into normalised feature vectors.

2.2.1 Preprocessing

Pre-emphasis

The high-frequency components in speech tend to contain less energy that those of lower frequency. Because these higher-frequency components are not less important, they need to be emphasised to ensure that the important information contained are adequately modelled. This emphasis is applied by the pre-emphasis filter with transfer function

$$H(z) = 1 - 0.98z^{-1}. \tag{2.1}$$

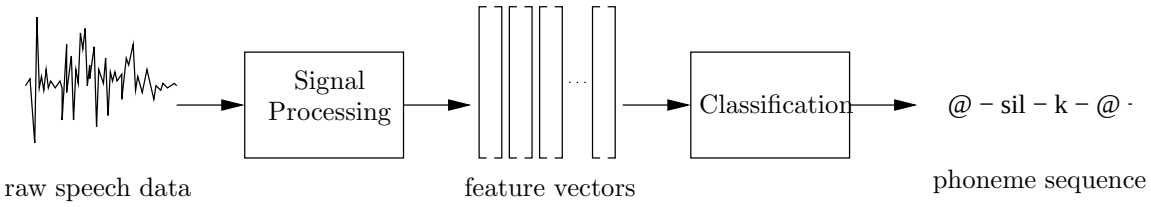


Figure 2.1: *Block diagram of a basic phoneme recognition system.*

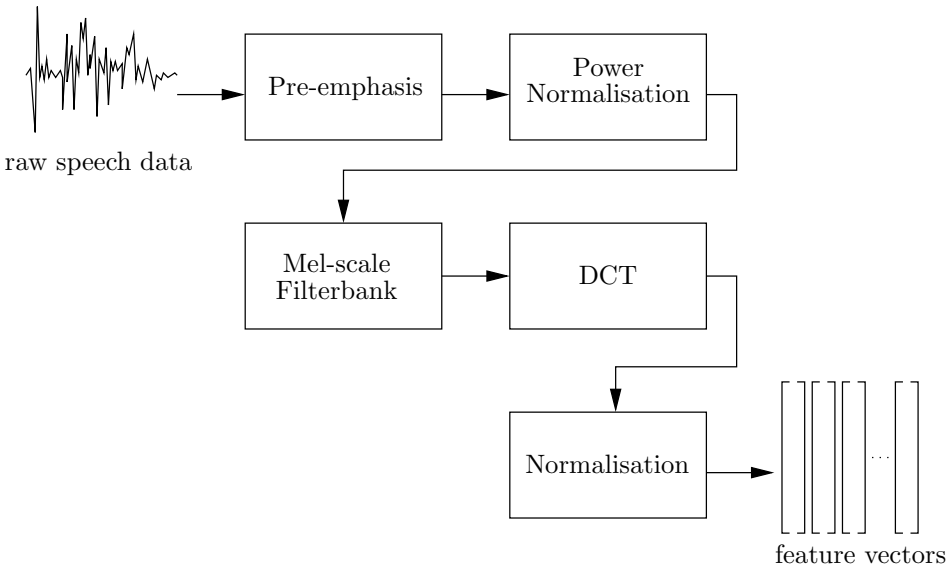


Figure 2.2: *Block diagram of signal processing, from raw speech to normalised feature vectors.*

Power Normalisation
The signal power for different recorded pieces of speech might not be equal. This might indicate a difference in the volume of the speech signal itself. To compensate for these power differences, we scale each speech signal to unity power.

2.2.2 Feature Extraction

After preprocessing the signal is ready for feature extraction. In our experiments Mel-scale Frequency Cepstral Coefficients (MFCCs) were used [9].

Frames of speech (of length 30ms) are taken and converted to the frequency domain using the Discrete Fourier Transform (DFT). Triangular overlapping windows are applied and we map to the Mel-scale using

$$Mel(f) = 1127.01048 \ln\left(1 + \frac{f}{700}\right). \tag{2.2}$$

The Mel-scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another. After the mapping the DCT is once again applied to the log of the Mel filter bank coefficients. The resulting cepstral (spectrum of a spectrum) coefficients are grouped into a feature vector.

2.2.3 Feature Normalisation

Channel Mean Subtraction
Channel effects also influence the features. These effects could include different microphones, landlines versus cellphone, distance from the microphone, background noise, etc. For better quality features, we also then need to compensate for these. Subtracting the mean of each cepstrum dimension per recording is a simple way to compensate for linear channel distortions.

Scaling
To reduce precision errors during calculation with a computer, the feature vectors are scaled to unity variance over each dimension.

Rate of change
Results are improved by not only taking into account the cepstral coefficients of a specific vector, but also the rate of change of these values. This is closely related to a derivative, but is much simpler to calculate. Each feature vector is basically concatenated with the four features directly before and after it. This results in a dimension nine times the original. Utilising this frame as a vector has the same effect as directly calculating the derivatives and contains all the information that a derivative can provide.

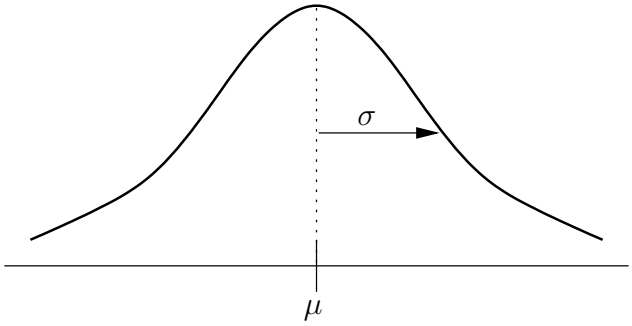


Figure 2.3: *One dimensional Gaussian PDF with mean μ and standard deviation σ.*

Dimension Reduction

After the previous post-processing techniques we are left with feature vectors of very high dimension. We need to find a way to reduces the feature vectors to a lower dimension, without losing any important information. Various linear transformation methods are available. In this case, Linear Discriminant Analysis (LDA) was used.

Linear Discriminant Analysis is a technique closely related to Principal Component Analysis [17] (also known as the Karhunen-Loéve Transform). Both apply a linear transform to achieve a lower target dimension, but where PCA chooses a combination of features in the direction of maximum variance, LDA chooses the linear combination that best separates the different classes. The mathematical detail is of no importance in the context of this thesis, but more detail can be found in [15].

2.3 Probability Density Function

Due to the fact that most random natural events assume the distribution of a Gaussian [21] PDF, it is the most popular of the basic density functions. More important, it leads to simple calculations. In Figure 2.3 we can see a typical Gaussian PDF with its mean μ and standard deviation σ. A one-dimensional Gaussian PDF is defined as

$$p(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-(x-\mu_x)^2/2\sigma_x^2} \tag{2.3}$$

with the standard deviation $\sigma_x > 0$ and mean $-\infty < \mu_x < \infty$.

In general, we deal with multivariate Gaussian PDFs, or in other words, a multi-dimensional Gaussian PDF. The multivariate Gaussian PDF is defined as

$$p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}, \tag{2.4}$$

with mean vector μ and covariance matrix Σ.

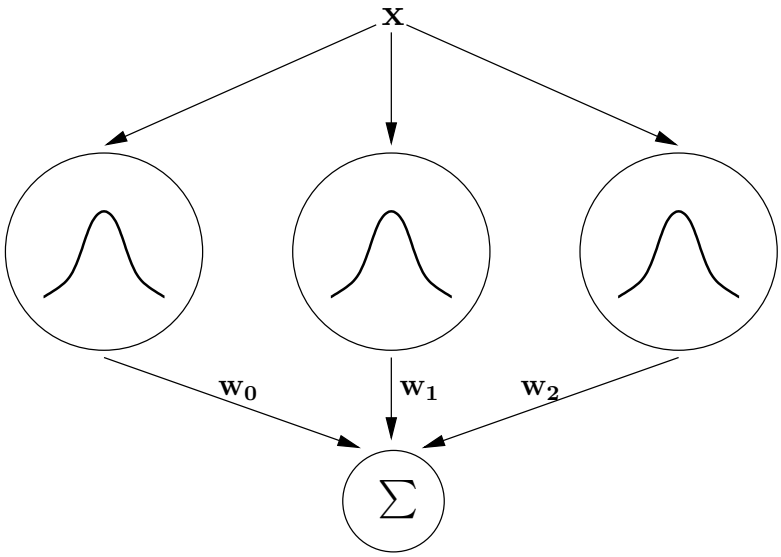


Figure 2.4: *Example of a three-component Gaussian Mixture Model with component weights w_0, w_1 and w_2.*

2.4 Mixture Models

Mixture Models are used when a single PDF cannot accurately model the information contained in the feature vectors. With data as complex as speech, mixtures with hundreds of components are typical. Gaussian Mixture Models (GMMs) are in essence a weighted sum of K component Gaussian PDFs with

$$p(x) = \sum_{k=1}^{K} w_k p_k(x) \tag{2.5}$$

The weights w_k satisfy the constraints $0 \leq w_k \leq 1$ and $\sum_{k=1}^{K} w_k = 1$.

Tree-Based Adaptive Gaussian Mixture Models (T-BAGMM) [10] provide a very accurate approximation of a standard GMM at a fraction of the training and testing time. This is achieved by modelling the GMM structure as a tree, using approximating nodes to generate a virtual GMM. A basic example can be seen in Figure 2.5.

2.5 Hidden Markov Models

A Hidden Markov Model (HMM) [22] is basically a state machine with states connected by transition probabilities. Each state contains a distribution function that can be a PDF, a Mixture Model or even another HMM. Figure 2.6 contains a basic five-state left-to-right HMM.

Several algorithms exist for HMM training, with the Viterbi algorithm being a popular choice. By iteratively finding the optimal path through the states, the transition probabilities

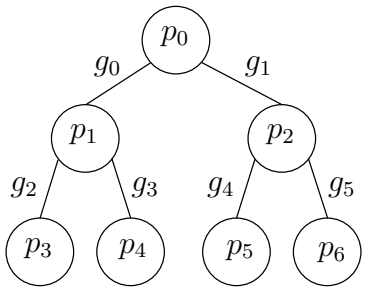


Figure 2.5: *Example of a Tree-Based Gaussian Mixture Model with seven nodes, each with weight g_i and Gaussian PDF $p_i = p(x|\lambda_i)$*

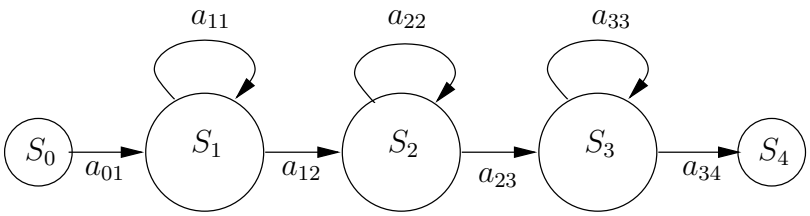


Figure 2.6: *Example of a five-state left-to-right Hidden Markov Model with start state S_0 and stop state S_4 and transition probabilities a_{ij}*

and the internal distribution functions can be re-estimated until the parameters stabilise.

2.6 A Basic Phoneme Model

Most phonemes do not stay constant over time and might change from beginning to end. The phoneme {Oi} as in b⟨*oy*⟩ is a good example. To model this phoneme, we split it into four states, $\texttt{Oi}_0$, $\texttt{Oi}_1$, $\texttt{Oi}_2$ and $\texttt{Oi}_3$. We use a four-state left-to-right Hidden Markov Model shown in Figure 2.7 with T-BAG Mixture Models as internal distributions to model this specific phoneme.

2.7 Summary

In this chapter the basics needed to understand the models and techniques used to implement a phoneme recogniser was introduced. Only the models and techniques used in our test systems were discussed, as other variations fall outside the scope of this thesis. We briefly showed how feature vectors are extracted from raw speech data and normalised. The Gaussian Probability Density Function is introduced next, with a summary of Mixture Models (specifically the T-BAGMM) following. A short introduction to Hidden Markov Models and a description on how the models fit together to form a phoneme model ends off the chapter.

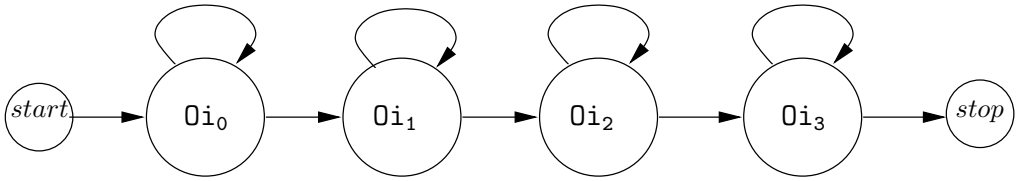


Figure 2.7: *The HMM model for the phoneme* {Oi}.

Chapter 3

Fusion from Hard Decisions

3.1 Introduction

This chapter covers fusion from hard decisions. The basic voting system is described in Section 3.2. Section 3.3 expands on the idea by introducing the weighted voting system. In the case where the decisions of some classifiers are known to be more trustworthy than that of others, the decisions of those classifiers can be given a larger weight. In Section 3.4 we introduce a classifier that uses decisions (class labels) and outputs constructed scores based on the characteristics of the specific phoneme. These characteristics are the phoneme class and the place of articulation.

3.2 Basic Voting

In the most basic voting system each classifier is assumed to have the same level of accuracy and is thus treated equally. One advantage of such a voting system is that no further training data is needed for the fusion. These voting techniques have been shown [13] to be a simple and effective method of combining classifier decisions. Various methods exist, with a few being plurality voting, majority voting, amendment voting, runoff voting and Condorcet counting. A brief description of each of these methods follow.

Plurality vote: This is the most basic and easily implemented voting method. Every voter has one vote that can be cast for any candidate. The candidate with the highest number of votes wins.

Majority vote: Every voter has one vote than can be cast for any candidate. A candidate can only win if it received a majority (more than half) of the votes. This method is more accurate than the plurality voting method, but in the worst case scenario where no candidate achieves a majority, no result is produced.

Amendment vote: This is an iterative majority voting method. Voting starts for the first two candidates. The winner is determined by majority voting and then pitted against the next candidate until only one candidate remains.

	class0	class1	class2
A	1		
B		1	
C	1		
D	1		
E			1
vote	3	1	1

Table 3.1: *Example of a plurality vote.*

Runoff vote: This is a two-step voting process. Two candidates are chosen in the first round by plurality voting. A majority vote in the second round determines the winner between the two.

Condorcet count : This method compares all candidates in pairwise plurality voting elections. One point is scored for the winner of each vote. The candidate with the highest number of points wins.

We decided on the simplest of voting methods, plurality voting. Table 3.1 shows an example of this voting system with five systems and three classes.

The only ambiguous factor is which class should be chosen if a tie occurs. A random choice can be made between the classes involved, or a trust factor can be included. If we assume for example that one system is more accurate than any other, the decision of that system should always be used as the deciding vote.

3.3 Weighted Voting

If we do not assume that all classifiers are equal and we want to reflect that in our voting system, we need to assign a fitting weight to each classifier. A weighted or confidence voting system assigns a weight to the vote of each voter. Various implementations exist, with a few being the pandemonium vote, the sum rule and the product rule.

Pandemonium vote: Every voter is given one vote that can be cast for any candidate. A vote is cast by assigning the confidence or weight as a vote. The candidate that received the vote with the highest confidence, wins. Thus after voting, only the winning voter and the confidence of its vote is known.

Sum rule: Every voter is given one vote than can be cast for any candidate. All the confidence values for each candidate is added and the candidate with the highest sum wins.

Product rule: Every voter is given one vote than can be cast for any candidate. The confidence values per candidate is multiplied and the candidate with the largest product confidence wins. In this method one very low value can greatly restrict a candidates chances of winning.

We decided on the sum rule. It retains the simplicity of plurality voting on which it is based, but is more robust than the pandemonium vote.

Our example in Table 3.2 assumes that system `A` and `B` is relatively equal in accuracy, `C` is a bit weaker and `D` and `E` the weakest. These weights indicate that more trust should be placed in the opinions of `A` and `B` than that of `C`, `D` and `E`.

	class0	class1	class2
A	5/20		
B		5/20	
C	4/20		
D		3/20	
E			3/20
vote	9/20	8/20	3/20

Table 3.2: *Example of a weighted vote.*

3.4 Voting With Knowledge of Phoneme Characteristics

In this section we introduce a new system that not only fuses the decisions of each classifier, but also takes into account certain characteristic of each phoneme. The vote of each classifier is not just a single class label, but a vector of constructed class scores. While a fusion of scores is the topic of the next chapter, the fact that these scores are generated from hard decisions makes for a better fit in the current one.

The two characteristics we investigate are the phoneme class and the place of articulation. Example phoneme classes are nasal sounds ({`m`}, {`n`}), fricatives ({`s`}, {`f`}) and open vowels ({`a`}, {`u`}). The place of articulation can be defined as the point of contact where an obstruction occurs in the vocal tract. For both vowels and consonants, we divided the place of articulation into three groups, namely front, middle and back. Examples for front consonants are {`b`}, {`m`} and {`p`}. The different phoneme class and place of articulation groups are listed in Appendix B.

In an ordinary voting situation, the labelled class is in essence assigned a score of *one* and the other classes scores of *zero*. This method build on that idea, but attempts to be more robust by assigning non-zero scores to the other classes as well. Each class model constructs

a score based on the labelled class. The constructed scores are generated according to the phoneme groups with the score levels chosen as follows (in order of highest to lowest):

- The actual labelled phoneme.
- Phonemes of the same class and in the same place of articulation.
- Phonemes of the same class.
- Phonemes in the same place of articulation.
- Phonemes not matching any group.

The rationale behind these phoneme groups is to take into account that for example, misclassification of an {a} as an {o} is *less wrong* than classification as an {s}.

	u	o	s	t	m
system0	1.0	0.7	0.1	0.1	0.1
system1	0.7	1.0	0.1	0.1	0.1
system2	0.1	0.1	0.1	0.1	1.0
system3	1.0	0.7	0.1	0.1	0.1
vote	2.8	2.5	0.4	0.4	1.3

Table 3.3: *Example of a vote with phoneme characteristics taken into account.*

Our example in Table 3.3 contains four classifiers, with phonemes {u}, {o}, {s}, {t} and {m}. {u} and {o} are in the same class and in the same place of articulation. {s} and {t} are in the same place of articulation. {m} is in a separate phoneme group and does not share a place of articulation.

The four systems classify the phonemes as {u}, {s}, {m} and {u}. For this example the scores were chosen as follows: 1.0 for the exact phoneme, 0.7 for the same class and place of articulation, 0.6 for the same class, 0.5 for the same place of articulation and 0.1 for phonemes not sharing any characteristic.

3.5 Summary

In this chapter we covered fusion from hard decisions. The technique of voting to determine the "best" final decision was introduced first, with a weighted voting system described next. We do not expect much from the performance of these techniques, but their trivial implementation makes it an attractive option. The fact that only the already classified class labels are used, makes it possible to fuse different kinds of classifiers, without having to compensate for scores that do not match types. Our last technique in this chapter accepts class labels as input, but outputs constructed scores for each class according to the phoneme

group and place of articulation of that phoneme. These constructed scores are then fused. The reasoning behind this is that we want to give a larger weight to phoneme classes that are “almost correct”, than those obviously “wrong”.

Chapter 4

Calibration with the Pool Adjacent Violators (PAV) Algorithm

4.1 Introduction

In the previous sections we assumed that the output scores of each classifier are equal in meaning. We assumed that if one classifier outputs a score in a certain range, the output of another classifier, with scores in the same range, should imply the same level of *certainty* by both. Even for classifiers of the same type, this is not always the case. Calibration techniques can be used to transform output scores, ensuring equality by the same metric. The Pool Adjacent Violators algorithm is one such technique.

4.2 Theory

The Pool Adjacent Violators (PAV) Algorithm [1][25][6][7] is used to find a non-decreasing mapping from scores into posterior probabilities. A block diagram of a PAV calibration system with a PAV curve for each class can be seen in Figure 4.1.

The simplicity of the PAV Algorithm is part of its appeal. All we need is a selection of target and non-target scores. A target score is defined as an output score from a class model, where the input feature vector belongs to that class. Likewise, a non-target score is defined as an output score from a class model when the input feature vector does not belong to that class. A summary of the PAV Algorithm is as follows:

- Choose the target and non-target scores.
- Assign a posterior probability of *one* to all target scores and *zero* to all non-target scores.
- Sort the vector of ones and zeroes according to the ascending order of the corresponding scores.

- Iteratively pool all adjacent values that violate monotonicity and replace them with the pool mean.

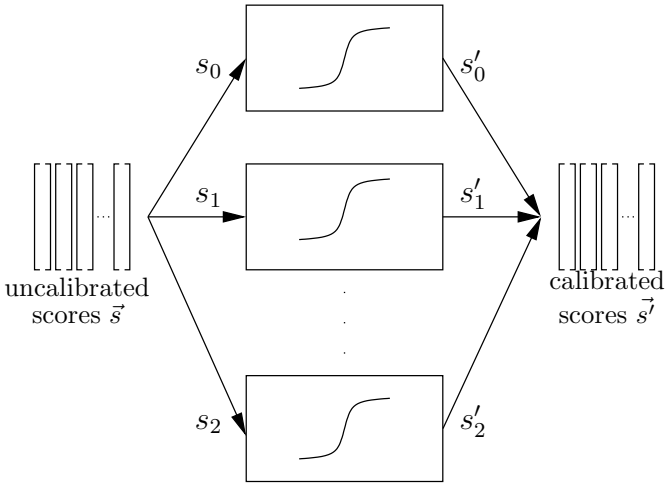


Figure 4.1: *Block diagram of PAV calibration with a PAV curve for each class*

As an example we have the following:

target scores:

$[-4.72 \quad -3.89 \quad -2.78 \quad -2.50 \quad -2.22 \quad -1.67 \quad -1.11 \quad -0.83 \quad -0.56]$

non-target scores:

$[-4.44 \quad -4.17 \quad -3.61 \quad -3.33 \quad -3.06 \quad -1.94 \quad -1.39 \quad -0.28]$

PAV input:

$[1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0]$

At each iteration of the algorithm the violating adjacent values (underlined) are pooled and averaged as follows:

1. $[\underline{1 \quad 0 \quad 0} \quad \underline{1 \quad 0 \quad 0 \quad 0} \quad 1 \quad 1 \quad \underline{1 \quad 0} \quad \underline{1 \quad 0} \quad 1 \quad 1 \quad \underline{1 \quad 0}]$
2. $\left[\frac{1}{3} \quad \frac{1}{3} \quad \underline{\frac{1}{3} \quad \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4}} \quad 1 \quad \underline{1 \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2}} \quad 1 \quad \underline{1 \quad \frac{1}{2} \quad \frac{1}{2}}\right]$
3. $\left[\frac{1}{3} \quad \underline{\frac{1}{3} \quad \frac{4}{15} \quad \frac{4}{15} \quad \frac{4}{15} \quad \frac{4}{15} \quad \frac{4}{15}} \quad 1 \quad \underline{\frac{3}{5} \quad \frac{3}{5} \quad \frac{3}{5} \quad \frac{3}{5} \quad \frac{3}{5}} \quad 1 \quad \underline{\frac{2}{3} \quad \frac{2}{3} \quad \frac{2}{3}}\right]$
4. $\left[\underline{\frac{1}{3} \quad \frac{5}{18} \quad \frac{5}{18} \quad \frac{5}{18} \quad \frac{5}{18} \quad \frac{5}{18} \quad \frac{5}{18}} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{3}{4} \quad \frac{3}{4} \quad \frac{3}{4} \quad \frac{3}{4}\right]$
5. $\left[\frac{2}{7} \quad \frac{2}{7} \quad \frac{2}{7} \quad \frac{2}{7} \quad \frac{2}{7} \quad \frac{2}{7} \quad \frac{2}{7} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{2}{3} \quad \frac{3}{4} \quad \frac{3}{4} \quad \frac{3}{4} \quad \frac{3}{4}\right]$

The resulting PAV curve for this example can be seen in Figure 4.2. As we can clearly see, the relatively low number of scores used for training results in a largely stair shaped curve. With large amounts of training data, the PAV curve approximates a sigmoidal shape.

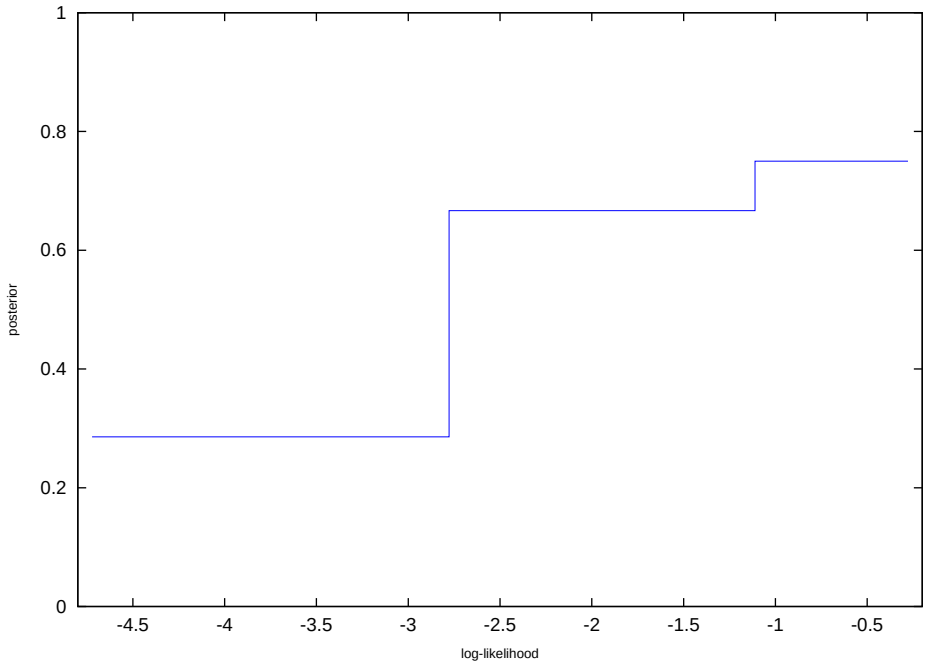


Figure 4.2: *Resulting PAV curve for the input [1 0 0 1 0 0 0 1 1 1 0 1 0 1 1 1 0].*

4.3 Experiments with synthetic data

MATLAB code for fusion and calibration (including the PAV algorithm) can be found at [3]. We used this toolkit for the following experiments with synthetic data.

Figure 4.3 shows two-dimensional synthetic data generated for three classes with a single Full Covariance Gaussian model for each class. The covariance matrices for the classes are equal and the chosen means are equally far apart.

$$\mu_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \quad \mu_3 = \begin{bmatrix} 3 \\ 3.7321 \end{bmatrix} \quad \text{and} \quad C_1 = C_2 = C_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

To simulate the situation where one class occurs much more frequently than any other, the data was generated in a $1:2:4$ proportion. Three datasets were generated: a training set, a test set and a set to be used for PAV training. Each dataset contains the following number of feature vectors for each class: 1000 for H_1, 2000 for H_2 and 4000 for H_3.

As we show in the following sections, the prior probability $P_i = P(H_i)$ plays an important role in PAV calibration. The priors according to class frequency are $P_1 = \frac{1}{7}$, $P_2 = \frac{2}{7}$ and $P_3 = \frac{4}{7}$.

Classifying with the models used to generate the data would result in ideal classification results. No information has been lost during training and no calibration errors have been made. Applying PAV curves to this classifier cannot enhance the results, but we can investigate the effect of various implementations of PAV.

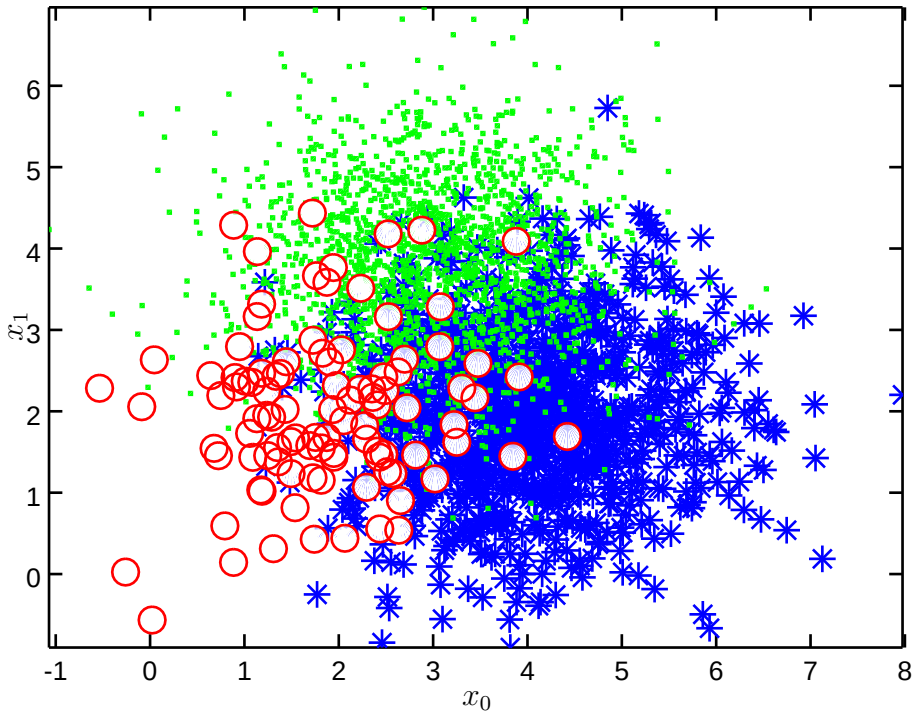


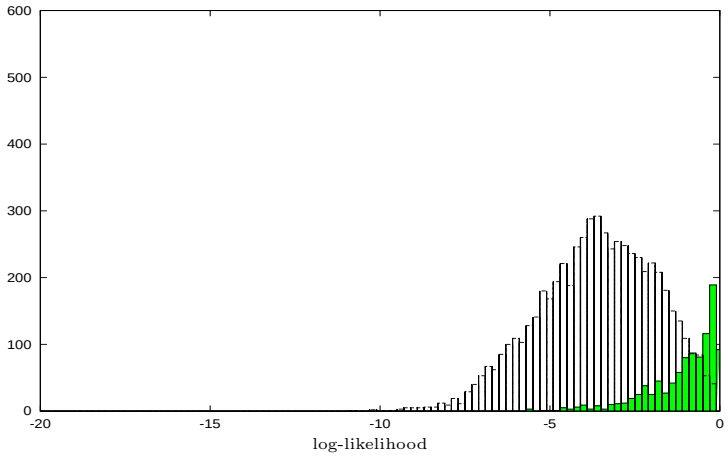
Figure 4.3: *The two-dimensional data for three synthetic classes.*

	total accuracy	H_1	H_2	H_3
with Prior	77.76%	51.70%	69.95%	88.18%
without Prior	74.97%	74.97%	74.97%	75.05%

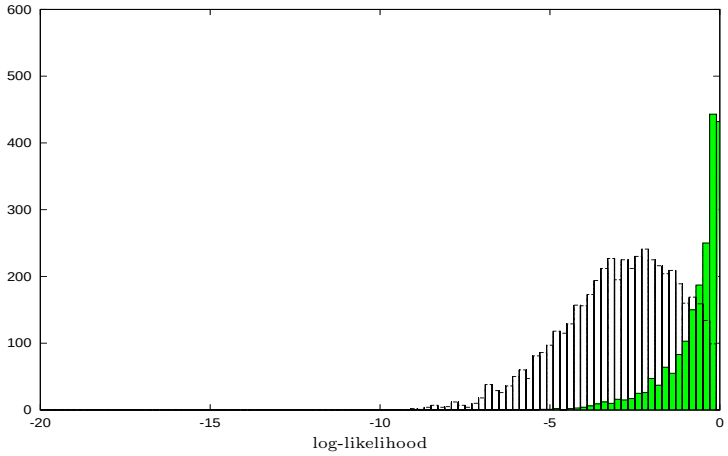
Table 4.1: *Total and class accuracies for classification with the ideal classifier, with and without the prior probabilities. As expected the classes are basically equal in accuracy, but when the priors are applied, the less frequently a class occurs in the data, the more it is suppressed in favour of the most frequent class.*

The classification results with our ideal classifier is shown in Table 4.1. When classifying with a flat prior, class accuracies are as expected, basically equal. Application of the actual prior also has the expected result of suppressing the least frequent classes in favour of the most frequent one. This raises the total classifier accuracy.

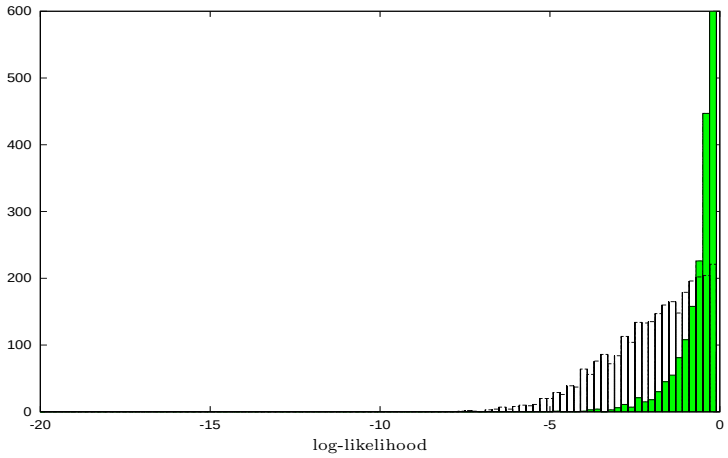
Figure 4.4 shows a histogram for all the target and non-target scores for each class. The large number of non-target scores for $class_1$ and $class_2$ can clearly be seen, as well as the large number of target scores for $class_3$. The proportion of target and non-target scores given as input to the PAV algorithm represents the prior probability for that specific class. Figure 4.5 shows the target and non-target scores for an equal number of target and non-target scores for each class. As we can see in this figure, no class is suppressed by a large number of non-target scores.



(a) $class_1$

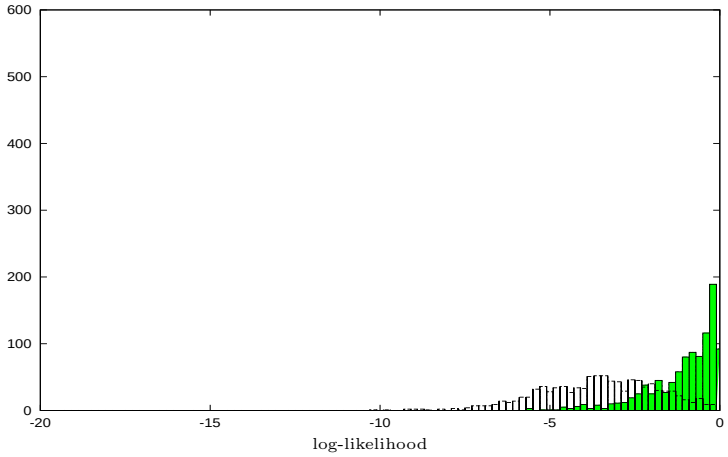


(b) $class_2$

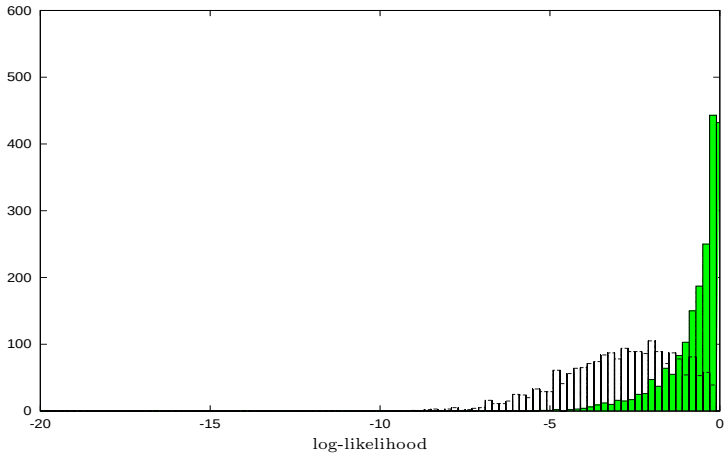


(c) $class_3$

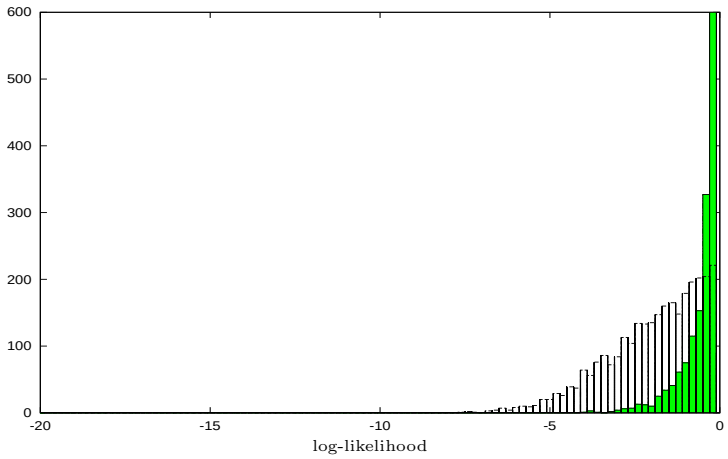
Figure 4.4: *Histograms of the target (solid bars) and non-target (outlined bars) scores for each class. The large number of target scores for $class_3$ is clearly visible.*



(a) $class_1$



(b) $class_2$



(c) $class_3$

Figure 4.5: *Histograms of an equal number of target (solid bars) and non-target (outlined bars) scores for each class.*

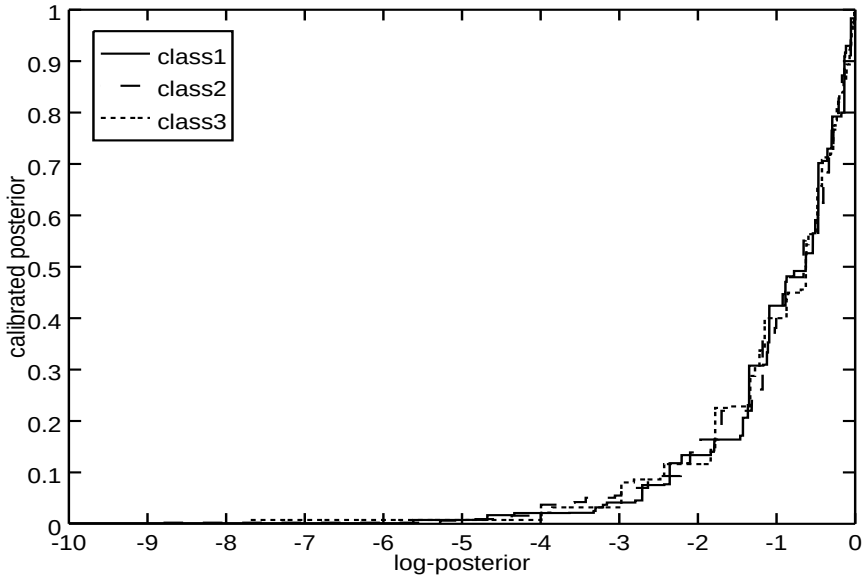


Figure 4.6: *The resulting PAV curve for each of the three classes, trained from posterior probabilities with the actual prior, for all target and non-target scores. The curves basically lie on a single line, with no bias for any class.*

For our first experiment we trained PAV curves from the posterior probabilities calculated with the actual prior, for all the target and non-target scores available to each class. The equation for the calculation of the posterior probabilities is as follows:

$$P(H_i|x) = \frac{P(x|H_i)P(H_i)}{\sum_{n=1}^{N} P(x|H_n)P(H_n)}, \tag{4.1}$$

for class H_i, feature x, prior $P(H_i)$ and likelihood $P(x|H_i)$. The resulting curves are shown in Figure 4.6, with a linear version shown in Figure 4.7. In the first figure we can see that the curves basically lie on a single line. The linear version shows that this line is almost an identity transform. This points to the fact that the scores are already ideal posterior probabilities based on the correct prior. The PAV calibration does not need to "fix" anything.

Classification results after this PAV mapping are shown in Table 4.2. The slight difference in accuracy is possibly the result of the stair shape of the PAV curves, which is especially prominent for the scarcest class, H_1.

Figure 4.8 shows the PAV curves for each class trained from the posterior probabilities with a flat prior, for all target and non-target scores available for each class. Here we can clearly see the effect that the proportion of target to non-target scores has on the PAV curves. A class with a higher proportion of target to non-target scores is mapped to higher posterior probabilities.

Table 4.3 shows the results from a mapping with these PAV curves. The result after the

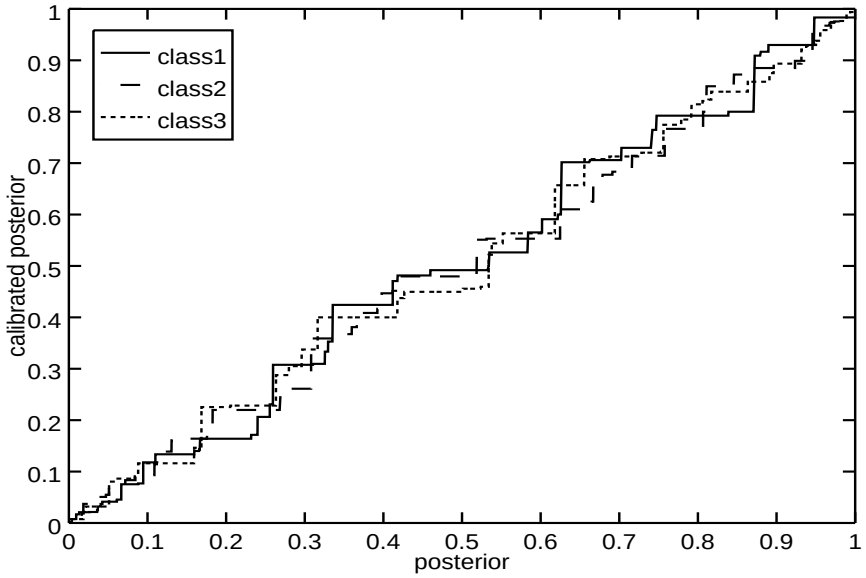


Figure 4.7: *The resulting PAV curve (linear) for each of the three classes, trained from posterior probabilities with the actual prior, for all target and non-target scores. The almost linear identity transform of each curve point to the fact that the scores are already ideal posterior probabilities.*

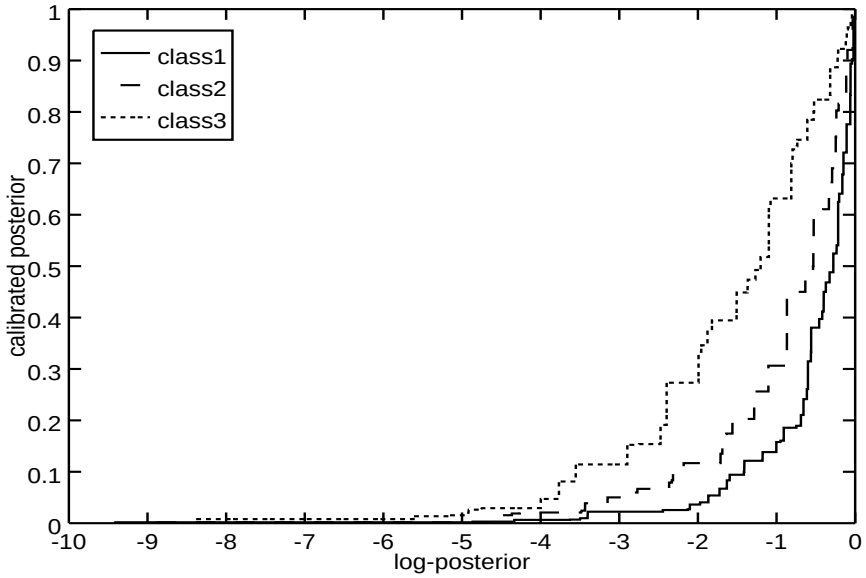


Figure 4.8: *The resulting PAV curve for each of the three classes, trained from posterior probabilities with a flat prior, for all target and non-target scores. The effect of the prior applied by PAV, due to the proportion of target to non-target scores, can clearly be seen.*

	total accuracy	H_1	H_2	H_3
posteriors (actual priors)	77.76%	51.70%	69.95%	88.18%
posteriors (actual priors) → PAV(all)	77.74%	55.10%	71.95%	86.30%

Table 4.2: *Total and class accuracies for classification with the ideal classifier with the prior probabilities, compared to that of a PAV mapping trained from posteriors calculated with the actual priors, for all target and non-target scores available for each class. The total accuracy for the two classifiers differ slightly.*

PAV calibration is close to that of the original after application of the actual prior. This shows that PAV can estimate the actual priors from the proportion of target to non-target scores.

	total accuracy	H_1	H_2	H_3
posteriors (actual priors)	77.76%	51.70%	69.95%	88.18%
posteriors (flat prior)	74.97%	73.90%	75.35%	75.05%
posteriors (flat prior) → PAV(all)	77.76%	58.70%	68.85%	86.98%

Table 4.3: *Total and class accuracies for classification with the ideal classifier with the actual prior and a flat prior, compared to that of a PAV mapping trained from posterior probabilities with a flat prior, for all target and non-target scores available for each class. PAV implicitly estimated priors from the proportion of the target to non-target scores, recovering accuracies close to those of the original priored case.*

The above experiments dealt with a mapping with PAV that includes the prior probability. In some cases we might want a flat priored PAV curve. This is achieved by supplying the algorithm with an equal number of target and non-target scores for each class.

Figure 4.9 shows the PAV curves trained from posterior probabilities with the actual prior, for an equal number of target and non-target scores. We can clearly see that PAV tries to reduce the impact of the applied prior by mapping the scarce classes to higher posterior probabilities than the more frequent class. The curves in Figure 4.10 were trained with a flat prior. In this case the scores for all classes are treated equally and we can see that the PAV mappings basically lie on a single curve.

In Table 4.4 we can see the classification results for calibration with the equal number of target and non-target scores. In the case where the prior was included in the scores used to train PAV, the algorithm succeeded in removing the prior, resulting in almost equal classification accuracies for all classes. The scarceness of data for the first class might have had an effect on the accuracy of the PAV curve itself. Some loss was shown for this class in both cases of the calibration.

The above experiments show the following:

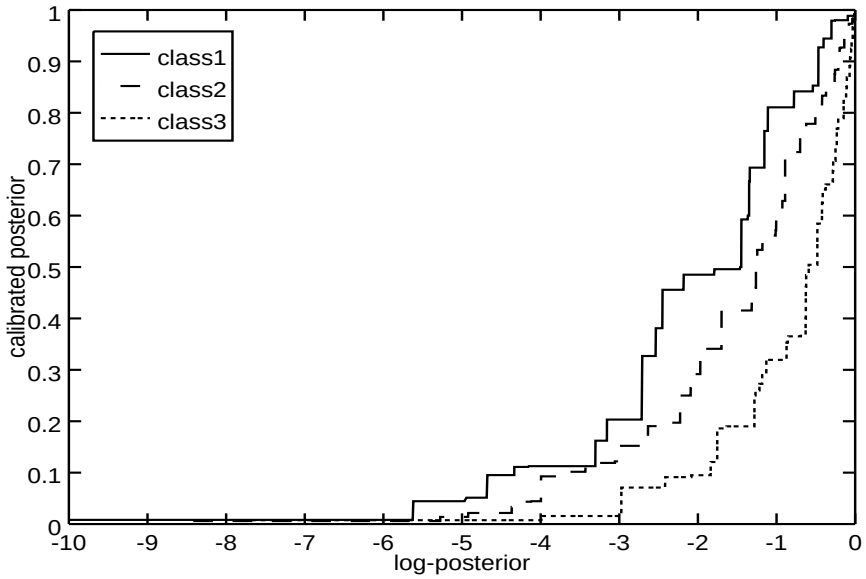


Figure 4.9: *The resulting PAV curve for each of the three classes, trained from posterior probabilities with the actual prior, for an equal number of target and non-target scores. The removal of the prior by PAV can clearly be seen in the lower mapped values for the more frequent class.*

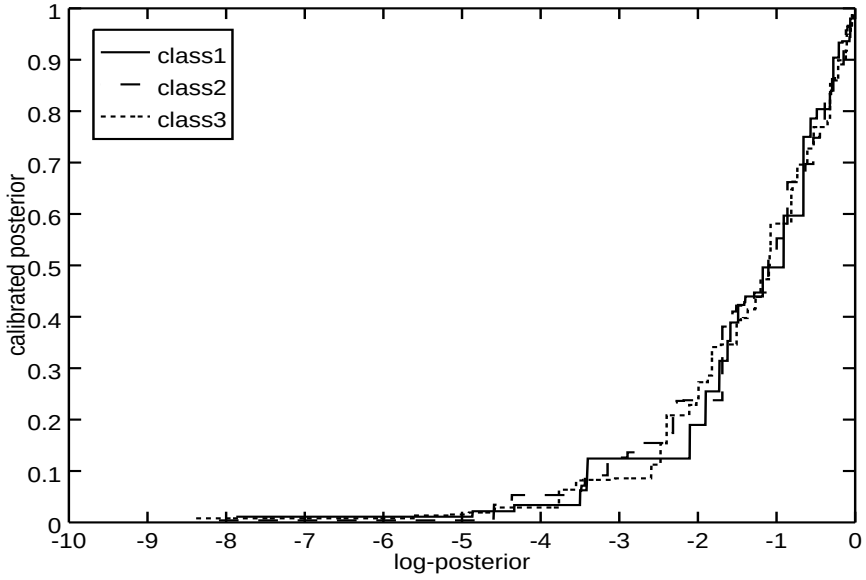


Figure 4.10: *The resulting PAV curve for each of the three classes, trained from posterior probabilities with a flat prior, for an equal number of target and non-target scores. No previously applied prior needs to be removed and the PAV mapping thus lie on a single curve.*

- The proportion of target to non-target scores plays an important role in PAV calibration.
- PAV can accurately estimate the prior from the proportion of target to non-target scores.
- A PAV calibration on an ideal system does not negatively influence the result if an adequate amount of training data is available.
- PAV can remove the effect of priors from scores.

	total accuracy	H_1	H_2	H_3
posteriors (actual priors)	77.76%	51.70%	69.95%	88.18%
posteriors (flat priors)	74.97%	73.90%	75.35%	75.05%
posteriors (actual priors) → PAV(1:1)	76.29%	68.20%	76.05%	78.43%
posteriors (flat priors) → PAV(1:1)	75.39%	72.50%	76.00%	75.80%

Table 4.4: *Total and class accuracies for classification with the ideal classifier with and without the prior probabilities, compared to that of calibration with PAV trained from log-likelihoods with and without priors, for an equal number of target and non-target scores. Both achieve basically the same results as the without-prior system. PAV succeeds in removing the prior from the with-prior case.*

Our next experiment investigates the effect of PAV on a non-ideal classifier. This is what one would typically expect from an actual recognition situation.

We perturb the classifier of the more frequent H_3 by adding an offset to the mean μ_3, moving the model further from H_1, but closer to H_2.

$$\mu_3' = \mu_3 + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

As we can see in Table 4.5, the accuracy of H_3 suffers from the offset to its mean. Even with the application of the prior probability, the inaccuracy of this class is still easily noticeable. The accuracy of H_1 is improved due to less interference from the now further away H_3 model.

	total accuracy	H_1	H_2	H_3
posteriors (actual priors)	73.13%	70.10%	72.05%	74.43%
posteriors (flat priors)	66.77%	82.10%	72.65%	60.00%

Table 4.5: *Total and class accuracies for classification with the perturbed classifier. The accuracy of H_3 greatly suffers from the offset to its mean. If we compare this to the after prior accuracy of that class with the ideal classifier (88.18%), we can see that application of the prior did not enhance the results by much.*

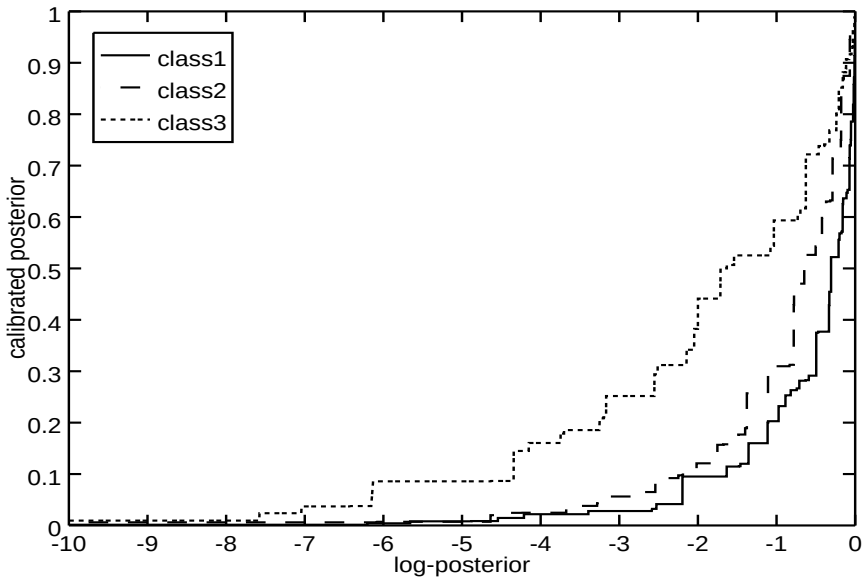


Figure 4.11: *The resulting PAV curve for each of the three classes of the perturbed classifier, trained from posterior probabilities with the actual prior, for the exact prior probability proportion of target to non-target scores. The low valued scores for H_3 are mapped higher to compensate for the erroneous offset in the model mean.*

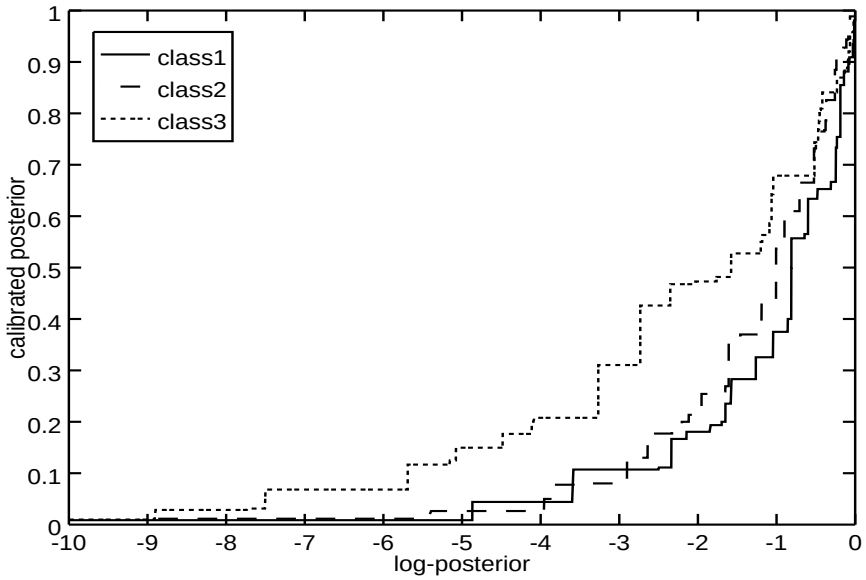


Figure 4.12: *The resulting PAV curve for each of the three classes of the perturbed classifier, trained from posterior probabilities with a flat prior, for an equal number of target and non-target scores. The lower scores for H_3 are mapped higher to compensate for the erroneous offset in the model mean.*

In Figures 4.11 and 4.12 we can see the trained PAV curves for this perturbed system. In both these figures we can see the PAV algorithm compensating for the offset to the mean of H_3 by mapping the lower scores to higher posterior probabilities than the curves for the other two classes. The effect of this can be seen in Table 4.6. Calibrating with PAV recovers some of the classification accuracy loss, by slightly suppressing the less frequent classes in favour of a large boost to the most frequent class.

	total accuracy	H_1	H_2	H_3
posteriors (actual priors)	73.13%	70.10%	72.05%	74.43%
posteriors (flat priors)	66.77%	82.10%	72.65%	60.00%
posteriors (actual priors) → PAV(all)	75.90%	55.90%	66.15%	85.78%
posteriors (flat priors) → PAV(1:1)	69.93%	78.80%	71.35%	67.00%

Table 4.6: *Total and class accuracies for classification with the perturbed classifier, in comparison to classification after PAV calibration. Total accuracy is improved by slightly suppressing the less frequent classes in favour of a large boost to the most frequent.*

4.4 Summary

This chapter introduced the concept of calibration and the Pool Adjacent Violators algorithm. This algorithm find a non-decreasing mapping from scores to posterior probabilities.

From the experiments with synthetic data we could see that a mapping with PAV does not have a negative influence on the output of an ideal classifier, but does show a positive effect on a classifier in need of calibration. When trained with an equal number of target and non-target scores, PAV has the ability to remove the prior from scores, unbiasing a classifier. In contrast, when trained with the number of target and non-target scores in proportion to the actual prior, PAV automatically includes the prior in the calibration mappings.

Chapter 5

Calibration and Fusion of Scores

5.1 Introduction

In contrast to Chapter 3, where we fused the hard decisions made by classifiers, this chapter deals with fusion of scores. A block diagram for a weighted score fusion system can be seen in Figure 5.1. Fusion with calibration is shown in Figure 5.2.

Basic weighted fusion is described in Section 5.2. Section 5.3 provides an introduction to C_{llr}, a metric that can be used to judge the quality of a classifier's decisions. This metric can by optimised with Linear Logistic Regression and used for both fusion and calibration.

5.2 Basic Weighted Fusion

A basic weighted fusion system can be seen in Figure 5.1. Each classifier outputs a vector of scores. These scores can be likelihoods, posterior probabilities, likelihood-ratios or any other meaningful metric.

An equally weighted combination of scores is the most basic method of fusing scores. With the following equation, the fused scores are the average of the scores from each classifier.

$$\vec{s} = \frac{1}{K} \sum_{k=1}^{K} \vec{s_k} \tag{5.1}$$

with scores $\vec{s}$ and number of classifiers K.

In the case where all classifiers are equal, the above method of averaging should achieve acceptable results. Different classifiers are unfortunately usually not equal. To compensate for this in a straightforward manner, a weight for each classifier can be chosen. The matter of how to choose the weight can be trivial or not, depending on whether further training is required. One basic way of choosing weights is to base it on the accuracy of each system. The equation for a weighted fusion is

$$\vec{s} = \sum_{k=1}^{K} w_k \vec{s_k} \tag{5.2}$$

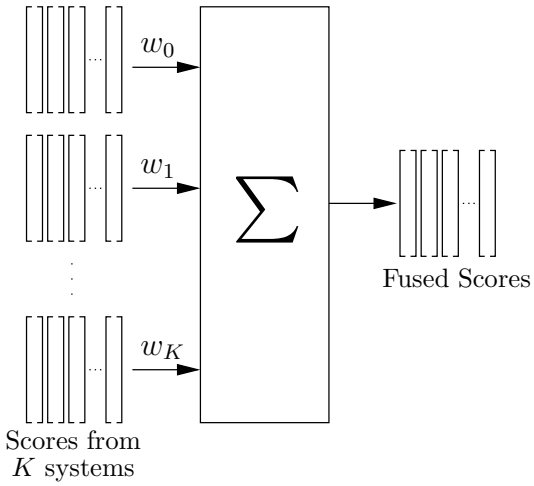


Figure 5.1: *Block diagram of a basic score fusion system.*

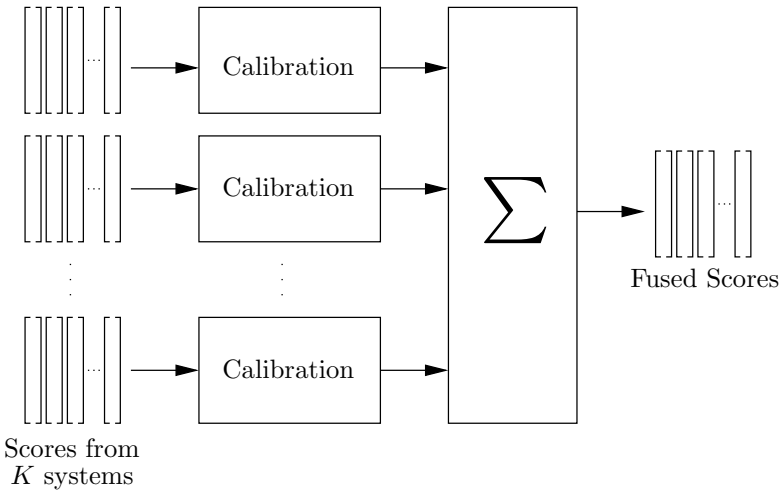


Figure 5.2: *Block diagram of a calibrated score fusion system.*

with scores $\vec{s}$, weights w and number of classifiers K.

5.3 Calibration and Fusion with C_{llr} and Linear Logistic Regression

This section introduces a metric used to measure the information delivered by a classifier and the optimisation of that metric for calibration.

Assume we have a system that classifies N classes, $H_1, \ldots, H_N$, with feature x as input and log-likelihood vector $\vec{\ell}$ as output. For each class the prior probability is known.

In [6] a metric is introduced for measurement and optimisation of the information content of log-likelihood vectors. This objective function is called multi-class C_{llr} and is defined as [4]

$$C_{llr} = -\frac{1}{T}\sum_{t=1}^{T} w_t \log_2 P_t \tag{5.3}$$

where P_t is the posterior probability of the true class of trial t, calculated from the log-likelihoods and a flat prior $P_{flat} = \frac{1}{N}$ as

$$P_t = P(H_{c(t)}|\vec{\ell}(x_t)) = \frac{e^{\ell_{c(t)}(x_t)}}{\sum_{j=1}^{N} e^{\ell_j(x_t)}}. \tag{5.4}$$

The function of the weight w_t is to normalise the class proportions in the evaluation trials.

$$w_t = \frac{P_{flat}}{Q_{c(t)}}, \qquad Q_i = \frac{\text{no. of trials for of class } H_i}{T}, \tag{5.5}$$

with $c(t)$ the true class of trial t.

The following interpretations can be made of values of C_{llr} (from [4]):

- $C_{llr} = 0$ only for perfect recognition. The posterior for the true class, $P_t = 1$ for every trial t.
- $C_{llr} = \infty$ if $P_t = 0$ for any trial t.
- $C_{llr} = \log_2 N$ is the reference value for a well calibrated, but useless recogniser. This recogniser extracts no information from the speech, but acknowledges this by by outputting equal log-likelihoods.
- $C_{llr} < \log_2 N$ indicates a useful recogniser. This recogniser can be expected to make decisions with a lower average cost than decisions based on the prior alone.

- $C_{llr} > \log_2 N$ indicates a bad recogniser. This recogniser can be expected to make decisions with a higher average cost than decisions based on the prior alone.

Information content measurement of recognisers has been covered above, but C_{llr} also has another use. It can be used as a logistic regression optimisation objective for calibrating and fusing recognisers.

The total amount of information loss to the system is represented by the raw unoptimised multi-class C_{llr}. This total loss can be defined as the sum of the *refinement* loss and the *calibration* loss.

$$C_{llr} = \textbf{refinement loss} + \textbf{calibration loss}. \tag{5.6}$$

The calibration loss is the maximum amount by which the total loss can be reduced through a linear transformation of the log-likelihood scores. In contrast, the refinement loss is represented by the optimised (minimum) value of C_{llr} and represents the loss that is inherent to the recogniser.

The mentioned calibration transformation is defined as

$$\vec{\ell}'(x_t) = \alpha\vec{\ell}(x_t) + \vec{\beta}, \tag{5.7}$$

with scalar α (scaling) and vector $\vec{\beta}$ (offset).

Instead of applying calibration separately for each recogniser as shown in the equation above, we can combine both calibration and fusion into one step as

$$\vec{\ell}'(x_t) = \sum_{k=1}^{K} \alpha_k \vec{\ell}_k(x_t) + \vec{\beta}, \tag{5.8}$$

with K recognisers, scaling constants α_k and vector $\vec{\beta}$.

The α_k and $\vec{\beta}$ parameters are determined by minimising C'_{llr}, based on the log-likelihood vectors $\vec{\ell}'(x_t)$. C'_{llr} is used as part the objective function for a logistic regression optimisation.

This regularised objective function is defined (from [5]) as

$$\mathcal{O}(\vec{\alpha}, \vec{\beta}) = \sum_{i=1}^{N} \frac{P_i}{\|\mathcal{C}_i\|} \sum_{t \in \mathcal{C}_i} \Big[-\log(\sigma_i(\vec{\ell}'_t + \vec{\pi})) \Big] + \epsilon \sum_{k=1}^{K} \alpha_k^2 + \lambda \sum_{i=1}^{N} \beta_i^2, \tag{5.9}$$

where

- $P_1, P_2, \ldots, P_N$ is the chosen prior class distribution
- $\vec{\pi} = [\log P_1, \log P_2, \ldots, \log P_N]'$ is the log prior
- $\mathcal{C}_i$ is the subset of data for class i
- $\sigma_i(\vec{x}) = e^{x_i}(\sum_{j=1}^{N} e^{x_j})^{-1}$ is the softmax function which calculates the posterior for class i

- $\vec{\ell}'_t = \sum_{k=1}^{K} \alpha_k \vec{\ell}_{kt} + \vec{\beta}$ is the fused and calibrated log-likelihood vector
- K is the number of systems to be fused
- $\vec{\ell}_{kt}$ is the uncalibrated log-likelihood vector that system k gives for trial t
- $\vec{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_K]'$ is a vector of system weights
- $\vec{\beta} = [\beta_1, \beta_2, \ldots, \beta_K]'$ is an offset vector
- $0 < \epsilon \ll 1$ and $0 < \lambda < \infty$ are regularisation constants which can be tuned. Increasing λ will force β to be small, while ϵ ensures that the objective function is strictly convex

The function $\mathcal{O}(\vec{\gamma})$ is convex and therefore has a unique minimum that can be found with an optimisation algorithm.

At first glance it might be difficult to relate Equation 5.3 and 5.9 to each other. The major difference between the two is that Equation 5.9 is in a more generalised form, allowing for priors other than the flat prior $P_{flat} = \frac{1}{N}$. The log prior $\vec{\pi}$ is thus included in the softmax function used to calculate the posterior probability.

The unique minimum can be found with a variety of numerical optimisers. Detail of these optimisers are not in the scope of this thesis, but we list a few possible candidates. Minka[18] compared these eight numerical optimisers for logistic regression:

- Newton's Method
- Coordinate Ascent
- Conjugate Gradient Ascent
- Fixed-Hessian Newton Method
- Quasi-Newton
- Iterative Scaling
- Modified Iterative Scaling
- Dual optimisation

with the conclusion that Conjugate Gradient Ascent and Quasi-Newton algorithms far outperform the other algorithms in terms of speed. The FoCal Multi-class [4] toolkit, that we used for our experiments, makes use of the Conjugate Gradient algorithm.

For the following example we used the same synthetic data generated for our PAV experiments in Chapter 4. The ideal classifier for the three synthetic classes (H_1, H_2 and H_3) contains a Gaussian model for each class with the following properties:

$$\mu_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \quad \mu_3 = \begin{bmatrix} 3 \\ 3.7321 \end{bmatrix} \quad \text{and} \quad C_1 = C_2 = C_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

We perturb the mean of the model for the second class to form the non-ideal classifier $\mathcal{T}_1$ and seperately perturb the mean for the third class to form the non-ideal classifier $\mathcal{T}_2$. The perturbations are given by

$$\mu_2' = \mu_2 + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

and

$$\mu_3' = \mu_3 + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}.$$

In Figure 5.3 we can see the calibration and refinement loss (as defined in Equation 5.6) of the various classifiers. As expected, the ideal classifier shows no calibration loss and the two perturbed classifiers show more unrecoverable refinement loss and some calibration loss. The FoCal fusion of these two classifiers is also shown, with no calibration loss and less refinement loss than either of the two individual classifiers.

The trained FoCal fusion parameters are listed in Table 5.1. From the negative value of the β_0 parameter we can clearly see the least frequent class being suppressed a bit more when training with the actual priors, that when using a flat prior.

	α_0	α_1	β_0	β_1	β_2
with flat priors	0.5150	0.4378	-0.4835	0.1492	0.3343
with actual priors	0.4919	0.5070	-0.5813	0.2176	0.3637

Table 5.1: *The FoCal trained α and β parameters. As expected when training with the actual priors, the scores from the least frequent class are suppressed more than those of the more frequent classes.*

Table 5.2 shows the classification accuracies for the ideal classifier, the individual $\mathcal{T}_1$ and $\mathcal{T}_2$ classifiers, a basic scores averaging fusion of $\mathcal{T}_1$ and $\mathcal{T}_2$ and FoCal fusions of $\mathcal{T}_1$ and $\mathcal{T}_2$. In the case of the ideal classifier, the classes achieve approximately the same classification accuracy. With the perturbed classifiers $\mathcal{T}_1$ and $\mathcal{T}_2$, the models for H_2 and H_3 were in effect moved closer to each other and further away from H_1. This results in an increase of classification accuracy for H_1, with major decreases for H_2 and H_3.

A basic score-averaging fusion results in a slight increase in accuracy, but still much lower than the original ideal system. The FoCal fusion on the other hand recovers almost all the loss introduced by our added offsets. Training the FoCal parameters with the actual priors resulted in slightly higher accuracies for the more frequent classes. If we compare the total accuracy of 74.59% of the FoCal fusion to the 68.80% of the uncalibrated fusion, the potential of this method is truly shown.

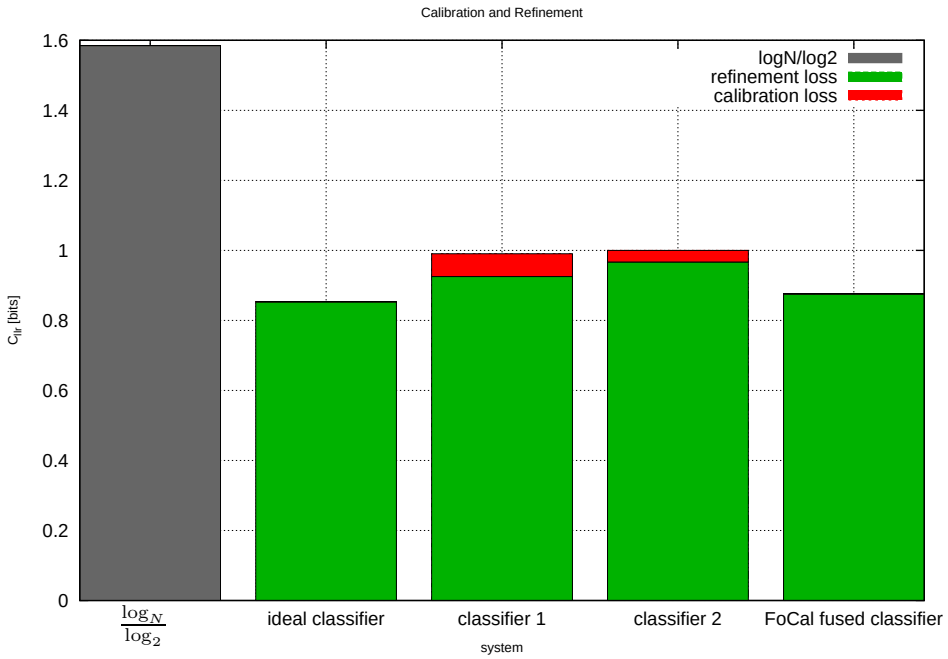


Figure 5.3: *The calibration and refinement loss for the ideal classifier, perturbed classifiers $\mathcal{T}_1$ and classifier $\mathcal{T}_2$ and the fusion of $\mathcal{T}_1$ and $\mathcal{T}_2$.*

	total accuracy	H_1	H_2	H_3
ideal classifier	74.76%	74.80%	75.05%	74.60%
$\mathcal{T}_1$	67.23%	81.40%	73.30%	60.65%
$\mathcal{T}_2$	67.29%	80.60%	62.45%	66.38%
uncalibrated equally weighted fusion	68.80%	83.90%	69.55%	64.65%
FoCal fusion with flat priors	74.09%	75.40%	74.55%	73.53%
FoCal fusion with actual priors	74.59%	74.30%	75.50%	74.20%

Table 5.2: *Total and class accuracies for classification with the ideal classifier, the perturbed classifiers $\mathcal{T}_1$ and $\mathcal{T}_2$, a basic score-averaging fusion of the two and the FoCal fusion with flat priors and the actual priors. The positive effect of the FoCal calibration and fusion can easily be seen in these results.*

5.4 Summary

This chapter introduced the concept of the fusion of classifier output scores. We covered a basic weighted score fusion first, commenting on equally weighted systems and one trivial way to calculate the weights for a non-equal system.

Our final section in this chapter introduced a metric C_{llr} that can be used to measure the quality of the information delivered by a classifier. By using C_{llr} as an objective function for an optimisation technique, we can calibrate the scores of an individual classifier, or calibrate and fuse the scores of multiple classifiers in a single step.

Chapter 6

The Bagging and Boosting Algorithms

6.1 Introduction

Bagging and Boosting are both techniques used to train a multitude of weak classifiers from subsets of a single dataset. Section 6.2 introduces the Bagging Algorithm. Boosting, specifically the AdaBoost algorithm and a multi-class variant thereof (SAMME), is described in Section 6.3.

6.2 Bagging

The acronym **bagging** stems from the words "**b**ootstrap **agg**regat**ing**" ([2]). Bootstrap samples are repeatedly taken from a single dataset, classifiers are trained on those subsets and the decisions of those classifiers are aggregated through use of voting or averaging.

Let $\mathcal{L}$ be a dataset containing data $\{(y_n, x_n),\, n = 1, \ldots, N\}$. We can train a weak classifier $\mathcal{T}(x)$ that assigns a label y to input feature x. The concept of a *weak* classifier can be defined as a basic classifier that is quick and easy to train, but only gives barely acceptable results. The number of aggregating classifiers to be trained is M. A summary of the Bagging Algorithm is as follows ([2]):

1. For $m = 1$ to M
   (a) Randomly draw, with replacement, N samples from $\mathcal{L}$ to form $\mathcal{L}^{(m)}$
   (b) Train classifier $\mathcal{T}^{(m)}(x)$ on $\mathcal{L}^{(m)}$.
2. Form classifier $\mathcal{T}(x)$ by fusing the decisions from $\mathcal{T}^{(m)}(x)$, for $m = 1 \ldots M$.

As the above summary shows, the Bagging algorithm is easy to grasp and implement. Some emphasis should be placed on the random drawing of the samples *with replacement*. This results in some samples not occurring in a specific subset, while others might occur multiple times. While the use of only one of the weak classifiers $\mathcal{T}^{(m)}(x)$ would result in substandard accuracies, a fusion of all should result in increased classifier stability and a decrease in variance.

6.3 Boosting

Boosting [14] and Bagging are related by the concept of using subsets of the training data to train a multitude of weak classifiers. They differ in the way in which these subsets are formed. Subsets for Boosting are not formed by randomly drawn samples, but contains all the samples of the original set, weighted according to classifier accuracy.

Various Boosting algorithms exist, with AdaBoost being one of the most popular ([26]). AdaBoost is an algorithm that adjusts adaptively to the errors introduced by a weak classifier, hence the name (**ada**ptable **boost**ing).

Starting out with equally weighted training examples, AdaBoost builds a classifier that produces class labels. The weight of each misclassified training data point is increased (boosted). The next classifier is built using the new weights. This is repeated for a number of iterations. 500 or 1000 classifiers are typical.

We once again have a dataset $\mathcal{L}$ containing data $\{(c_n, x_n),\, n = 1, \ldots, N\}$, with c_n the true class of trial x_n. A weak classifier $\mathcal{T}(x)$, that assigns a label y to input feature x, can be trained using the weights $w_n = \frac{1}{N}$, $n = 1, \ldots, N$. The AdaBoost algorithm for training M classifiers is then as follows:

1. Initialise the observation weights $w_n = \frac{1}{N}$, $n = 1, \ldots, N$.

2. For $m = 1$ to M

   (a) Fit a classifier $\mathcal{T}^{(m)}(x)$ to the training data using weights w_n.

   (b) Calculate

   $$err^{(m)} = \sum_{n=1}^{N} w_n \mathbb{I}\left(c_n \neq \mathcal{T}^{(m)}(x_n)\right) / \sum_{n=1}^{N} w_n,$$

   with $\mathbb{I}(\ldots)$ the logical operator outputting one if true and zero if false

   (c) Calculate

   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}}$$

   (d) Set

   $$w_n \leftarrow w_n \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_n \neq \mathcal{T}^{(m)}(x_n)\right)\right),\; n = 1, 2, \ldots, N$$

   (e) Re-normalise w_n

3. Output

   $$C(x) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}\left(\mathcal{T}^{(m)}(x) = k\right)$$

A restriction on the effectiveness of the AdaBoost algorithm is the size of the error $err^{(m)}$ of each weak classifier. In step (2c) we can see that if $err^{(m)} > 0.5$, $\alpha^{(m)}$ will be negative, thus resulting in weights being updated in the wrong direction. In the two-class case, where AdaBoost has been shown to be very successful, this restriction is not very harsh. For $err^{(m)}$ to be larger than 0.5, a weak classifier has to do worse than random guessing.

In the case where the number of classes K is larger than two, an accuracy of 0.5 will be much harder to achieve. The accuracy for randomly classifying K classes is $\frac{1}{K}$, a value that for large values of K, is much less than the allowed 0.5.

In order to lighten this restriction a new multi-class AdaBoost algorithm was proposed in [26]. This new algorithm is referred to as SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function) and is summarised as follows:

1. Initialise observation weights $w_n = \frac{1}{N}$, $n = 1, \ldots, N$

2. For $m = 1$ to M

   (a) Fit a classifier $\mathcal{T}^{(m)}(x)$ to the training data using weights w_n.

   (b) Calculate

   $$err^{(m)} = \sum_{n=1}^{N} w_n \mathbb{I}\left(c_n \neq \mathcal{T}^{(m)}(x_n)\right) / \sum_{n=1}^{N} w_n$$

   (c) Calculate

   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1)$$

   (d) Set

   $$w_n \leftarrow w_n \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_n \neq \mathcal{T}^{(m)}(x_n)\right)\right), \; n = 1, 2, \ldots, N$$

   (e) Re-normalise w_n

3. Output

   $$C(x) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}\left(\mathcal{T}^{(m)}(x) = k\right)$$

Except for the difference in step (2c), the algorithm SAMME is exactly the same as AdaBoost. The addition of the extra term $(K - 1)$ results in a system much more stable for classifiers where $K > 2$. In contrast to AdaBoost that requires $(1 - err^{(m)}) > \frac{1}{2}$, SAMME is less strict with $(1 - err^{(m)}) > \frac{1}{K}$ for $\alpha^{(m)}$ to be positive. This restriction is easier to achieve and scales well for larger K. This again implies that the weak classifier should at least do better than guessing.

6.4 Summary

This chapter introduced the Bagging and Boosting algorithms. Both of these algorithms train and fuse multiple classifiers from subsets of the training data. The accuracy of each of these classifiers is much lower than a classifier one would usually train, therefore they are referred to as *weak* classifiers. A couple of hundred of these classifiers are typically trained and combined. This large number offsets the low accuracy of each individual classifier.

We describe the Bagging algorithm first, commenting on the way subsets are formed by randomly drawing with replacement. Different classifiers are therefore not trained on exactly the same dataset, resulting in classifiers that are better with some features than others.

Where the Bagging algorithm formed subsets by random drawing, Boosting assigns a weight to each feature vector in the dataset. Features that were misclassified by the classifier trained in the previous iteration are weighed more heavily. This should result in a classifier that achieves better results on the features that the previous classifier struggled with. AdaBoost is the most widely used Boosting algorithm and has been known to achieve very high accuracies. Unfortunately for the algorithm to remain stable, the error per classifier has to be less than 0.5. For two-class problems this restriction is easy to adhere to, as 0.5 is the error achieved by random guessing. Higher class classification has a much lower guessing accuracy and is not that suitable for AdaBoost. An extension to Adaboost called SAMME has been proposed and reduced the error restriction to less than $\frac{1}{K}$, for K classes. This restriction is once again easy to adhere to.

Chapter 7

Experimental Results

7.1 Introduction

This chapter contains a description of our test systems and the results of the various experiments.

Section 7.2 describes the African Speech Technology (AST) Speech Corpus used for the training of our base systems, fusers and calibration techniques. In Section 7.3 we explain the basic topology of our fusion system. The statistical models used for our phoneme recognisers are also briefly mentioned. We introduce the concept of class reduction with the use of Bayes' Theorem in Section 7.4. Before the results of any fusion experiments, we first discuss the accuracy of our base systems in Section 7.5.

In Section 7.6 we experiment with basic voting systems and also experiment with our technique that makes use of phoneme characteristics. Section 7.8 details our experiments with the Pool Adjacent Violators (PAV) algorithm. We cover the individual system results after calibration and the results of calibrated fused systems.

Our use of the information metric C_{llr} and Linear Logistic Regression (LLR) is shown in Section 7.9. We use the FoCal Multi-class toolkit [4] to calculate the calibration and refinement loss for each classifier. The toolkit is also used to calibrate and fuse the classifiers, with two separate experiments. In the first we calibrate and fuse in one single step, while in the second experiment we calibrate each classifier individually before fusion.

Section 7.10 describes our experiments with the Bagging and Boosting algorithms. We investigate the use of different model types as weak classifiers and also the number of those classifiers to be trained.

7.2 The African Speech Technology Speech Corpus

The African Speech Technology (AST) Project [11, 23] recorded speech data for five South African languages, namely English, Afrikaans, Xhosa, Zulu and Southern Sotho. For our experiments, we only focused on the English dataset.

Speech variation in South African English is traditionally culturally bound and in order

to make provision for these varieties, subsets were chosen based on different accent groups by mother-tongue and non-mother-tongue speakers. We used these accent group subsets for our testing system. The accents groups (and their associated two letter acronyms as defined in [23]) are Afrikaans English (`AE`), Black English (`BE`), Coloured English (`CE`), English English (`EE`) and Asian English (`IE`).

Volunteers, each with a given text sheet in his/her language, were asked to call into a phone system. The sheets contained digit strings, numbers, spelt words, place names and phonetically balanced sentences. Volunteers also had to provide information such as the type of telephone used (landline or cellphone), their ages and the town where they attended school. The recruiters tried to enlist an equal number of male and female volunteers. A wide spread of caller ages, between twenty and sixty, were included.

In total the English database contains about 6 hours of transcribed speech per accent group.

The database is divided into *training* and *testing* subsets. The *training* data was used to train the original `AE`, `BE`, `CE`, `EE`, `IE` and `ALL` (all data used for training) systems.

7.3 The Basic Fusion System

For our experiments we use six systems (`AE`, `BE`, `CE`, `EE`, `IE` and `ALL`), the first five trained on their respective dialect groups and one system trained on all dialects. The main goal is to fuse those five specialist systems (as shown in Figure 7.1) and compare the results to those of the generalised one (as shown in Figure 7.2). Whether or not a fusion of systems that are very accurate on their own specific accent, but weaker on others, will do better than the generalised one, should be an interesting experiment.

Some investigation was done into the possibility of fusing the output phoneme strings of the systems, but it was decided to rather fuse at a frame-by-frame basis, before any classification is done. One issue with fusing output phoneme strings is that due to differing phoneme lengths, these sequences do not exactly match in time.

Each system consists of HMMs with a state for each phase of a specific phoneme as shown in Figure 2.7 in Chapter 2.6. The output distribution function for each state is a T-BAG Mixture Model (Chapter 2.4). The internal workings of each classifier is not relevant to this thesis, as we only deal with the output scores (in this case, log-likelihoods).

We do not fuse the outputs of these HMMs, but instead, we use the scores from the internal classifiers themselves. For each input vector, a score is generated from each state of each HMM. While the experiments in this chapter only deals with the accuracies of these scores, it is possible to use a fusion of the internal classifiers as the distribution function for the states of a new HMM.

Our phoneme set consists of 44 phonemes shown in Appendix A. Almost all of these consist of more than one state, which results in 137 classes to be classified. Experiments are done for both the 137-class case and where the number of classes is once again reduced to

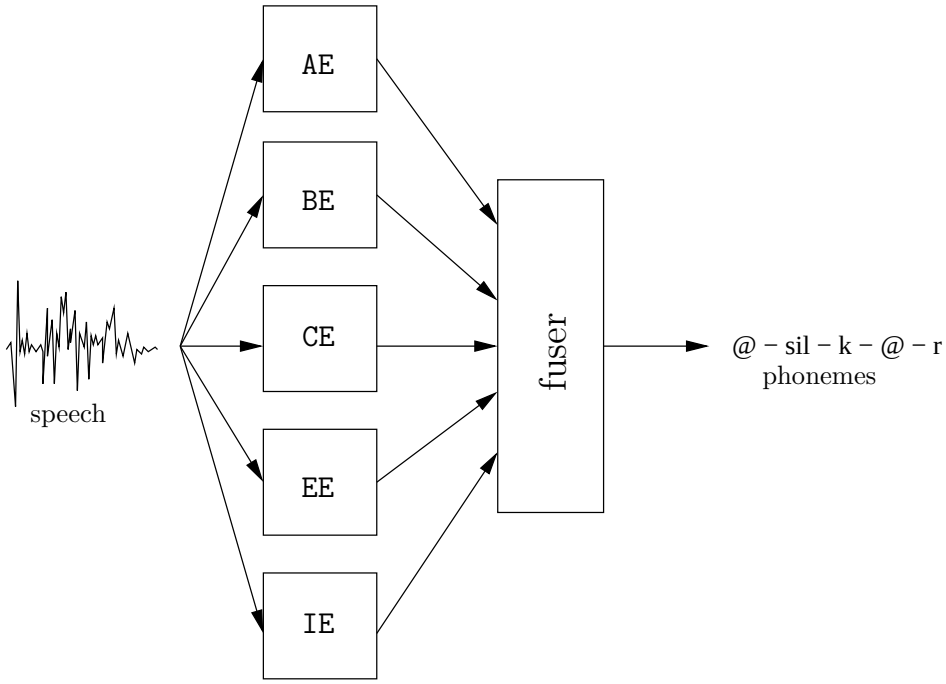


Figure 7.1: *Block diagram for our fusion test system.*

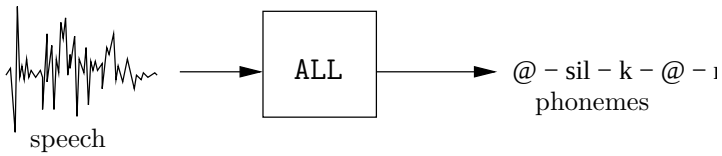


Figure 7.2: *Block diagram for our comparative test system.*

44 as will be explained in the next section.

7.4 Reducing 137 Phoneme HMM State Classes to 44 Phoneme Classes

As mentioned in Section 2.6, each phoneme was modelled according to the individual phoneme states. All these states result in a large number of classes, 137 in our case, that can be the cause of various difficulties during training and classification. More training data is needed. Memory usage is higher. Accuracies are much lower. We tried to compensate for these problems by reducing the number of classes to the original 44 phonemes.

For example, to accurately model the different transitions and sounds in the phoneme `Oi`, our model consists of four states. As classification output we thus receive four likelihood scores, $P(x|\texttt{Oi}_0)$, $P(x|\texttt{Oi}_1)$, $P(x|\texttt{Oi}_2)$ and $P(x|\texttt{Oi}_3)$. These likelihood scores can not be combined as is, but can be easily added if converted to posterior probabilities.

From Bayes' Theorem [21] the posterior probability is

$$P(H_i|x) = \frac{P(x|H_i)P(H_i)}{\sum_{n=1}^{N} P(x|H_n)P(H_n)}, \tag{7.1}$$

for class H_i, feature x, prior $P(H_i)$ and likelihood $P(x|H_i)$.

Before we can do the mapping with Bayes' Theorem, we need to choose the *priors*. One way is to use the frequency of each phoneme in the database as a prior. Even though this is valid and logical, we decided against this method. In our database a phoneme such as {`sil`} (the silence phoneme) occurs about 20 000 times, while a scarce phoneme such as {`Z`} (as in the word {*g*}*enre*), occurs only about 10 times. If we chose our prior according to these frequencies of occurrence, the scarce phonemes would be completely dominated by the frequent phonemes. Rather than bias the mapping in favour of the frequent phonemes, each phoneme is weighted equally, with a prior of $\frac{1}{44}$. For each phoneme the $\frac{1}{44}$ prior is then split according to the number of phoneme states.

{`sil`} for example has only one state and thus gets a prior of $\frac{1}{44}$. {`Oi`} consists of four states, each assigned a prior of $\frac{1}{176}$.

For the phoneme {`Oi`} we have

$$P(\texttt{Oi}_\texttt{k}|x) = \frac{P(x|\texttt{Oi}_\texttt{k})P(\texttt{Oi}_\texttt{k})}{\sum_{n=1}^{N} P(x|H_n)P(H_n)}, \qquad k = 0 \ldots 3$$

with the priors

$$P(\texttt{Oi}_\texttt{k}) = \frac{1}{176}.$$

The posterior probabilities of the phoneme states can then be added to form a single posterior probability for the entire phoneme.

$$P(\texttt{Oi}|x) = \sum_{k=0}^{K} P(\texttt{Oi}_\texttt{k}|x). \tag{7.2}$$

This single posterior probability score still contains all the information of the four separate likelihoods, but in effect decreases the amount of training data required and memory usage for our calibration and fusion techniques.

7.5 Individual System Accuracy

Before any fusion or calibration methods can be investigated, we need a reference to compare against. We use single-frame classification accuracy as our metric. Table 7.1 shows the accuracy of the original systems for the 137-class case. We can clearly see that the system trained on all the data is vastly superior.

AE	39.92%
BE	33.65%
CE	38.72%
EE	40.49%
IE	37.13%
ALL	47.75%

Table 7.1: *The accuracy of individual systems for the original 137 class case.*

Reducing to 44 classes with Bayes' Theorem and adding the states together, as in the previous section, results in the accuracies in Table 7.2. Please note that these accuracies can not be compared with the accuracies in the 137 class case, as we are not classifying the same classes. Higher accuracies are expected due to the lower number of classes.

AE	45.59%
BE	39.78%
CE	44.66%
EE	45.83%
IE	43.35%
ALL	50.30%

Table 7.2: *The accuracy of individual systems for the 44 class case.*

Due to the fact that we needed to map to posterior probabilities before reducing the number of classes, we were unable to compare the accuracy of classifying by likelihoods with that of classifying by posteriors for the 44 class case. This test can be performed with the original 137 classes. Table 7.3 shows the accuracy of the systems with all 137 classes after a mapping to posterior probabilities with Bayes' Theorem, using the prior probabilities as previously chosen. These results can be compared with those in Table 7.1 and a small increase in accuracy is observed.

AE	41.51%
BE	35.51%
CE	40.31%
EE	41.94%
IE	39.25%
ALL	49.44%

Table 7.3: *The accuracy of individual systems for the 137 class case with mapping to posterior probabilities.*

7.6 Hard Decision Fusion

This section covers our experiments using the techniques introduced in Chapter 3.

The most basic of our fusion techniques is the plurality vote. An equal vote is assigned to the decision of each classifier. The following accuracies (Table 7.4) were achieved in the original 137 and 44 class cases.

Equally weighted voting system (137 classes)	48.53%
Equally weighted voting system (44 classes)	53.58%

Table 7.4: *The accuracies for an equally weighted voting systems (137 and 44 classes). The accuracy for our baseline* `ALL` *classifier is 47.75% for 137 classes and 50.30% for 44 classes. An increase in accuracy can by seen for the 44 class classification after voting.*

The equally weighted system does show a small increase in classification accuracy. After our tests in Section 7.5 we know that some systems perform much better than others. We might need to incorporate the accuracies of these systems into the weight of their votes. The difference in accuracy between the `BE` and the `EE` classifiers for example are significantly large to warrant such a method. We test the classifiers on a previously unused subset of data and use the normalised accuracies achieved as the weight for each individual system. The following results in Table 7.5 were achieved.

Voting system, weighted according to classifier accuracy (137 classes)	46.87%
Voting system, weighted according to classifier accuracy (44 classes)	52.10%

Table 7.5: *The accuracies for a weighted voting system (137 and 44 classes), with weights chosen according to classifier accuracy. The accuracy for our baseline* `ALL` *classifier is 47.75% for 137 classes and 50.30% for 44 classes. A small increase in accuracy can be seen for the 44 class classification after fusion.*

These results (Table 7.5) are less accurate than that of the previous experiment. A possible explanation for this is that by weighing each system according to the total classification

accuracy, we might be reducing the opinion of each expert classifier for their specific group. For example, the vote of the `AE` system for data from its accent group, should probably not be weighted less than the `EE` system.

Our next experiments use the technique introduced in Chapter 3.4. Phonemes are divided into groups according to the phoneme class (nasal, fricative etc.) and the place of articulation (front, middle and back) as can be seen in Appendix B. After some discussion [12] and experimentation, the values for the constructed scores were chosen as shown in Table 7.6.

exact same phoneme	1
same phoneme class and place of articulation	$e^{-\frac{1}{6}}$
same phoneme class	$e^{-\frac{2}{6}}$
same place of articulation	$e^{-\frac{4}{6}}$
none of the above	$e^{-\frac{5}{6}}$

Table 7.6: *The constructed scores chosen for phoneme groups and place of articulation.*

The results in Table 7.7 does not show much of an improvement. Obtaining the constructed scores by training or optimisation methods might result in better accuracies than choosing by hand and should be considered for further investigation.

Voting with phoneme classes and place of articulation (137 classes)	45.94%
Voting with phoneme classes and place of articulation (44 classes)	51.68%

Table 7.7: *The accuracies for a voting system (137 and 44 classes), with knowledge of phoneme class and place of articulation. The accuracy for our baseline* `ALL` *classifier is 47.75% for 137 classes and 50.30% for 44 classes. A very small increase in accuracy can be seen for the 44 class classification.*

7.7 Basic Score Fusion

As mentioned in Chapter 5.2, the most basic way to fuse the output scores of classifiers is adding them together.

Assigning an equal weight to each classifier leads to a fusion system where the opinion (scores) of each system gets an equal influence on the final decision. The results of such a fusion is shown in Table 7.8.

There is a slight improvement in the 44 class case, but when classifying 137 classes the original `ALL` classifier still outperforms the fused score version.

In the next experiment we weigh the output scores of each classifier with its accuracy (determined in Section 7.6). The results are shown in Table 7.9.

Once again we see some improvement over the 50.30% achieved in the 44 class case, but no improvement over the 49.44% achieved with 137 classes.

Equally weighted score fusion (137 classes)	45.43%
Equally weighted score fusion (44 classes)	53.98%

Table 7.8: *The accuracies for an equally weighted score fusion system (137 and 44 classes). The accuracy for our baseline* `ALL` *classifier is 47.75% for 137 classes and 50.30% for 44 classes. A increase in accuracy can be seen for the 44 class classification after fusion.*

Weighted score fusion (137 classes)	45.45%
Weighted score fusion (44 classes)	53.95%

Table 7.9: *The accuracies for a weighted (according to classifier accuracy) score fusion system (137 and 44 classes). The accuracy for our baseline* `ALL` *classifier is 47.75% for 137 classes and 50.30% for 44 classes. A noticeable increase can be seen for the 44 class classification after fusion.*

Some closer investigation into the results in this section leads us to believe that these fusion techniques struggle more with the 137 class case than with 44 classes. One reason for this might be the lower classification accuracies found in systems with a large number of classes.

7.8 Calibration with Pool Adjacent Violators

For our experiments in this section we implemented PAV calibrators as discussed in Chapter 4. One PAV curve was trained for each class of each classifier.

7.8.1 Equal number of target and non-target scores

In this first experiment we ignore the differences in the frequencies of occurrence of the phonemes and treat all phonemes as equal by supplying an equal number of target and non-target scores to the PAV algorithm. Figure 7.3 shows the trained PAV curves for the phonemes {`sil`} (the most frequent phoneme), {`Z`} (one of the least frequent phonemes) and {`i`} (a typically occurring phoneme). Figure 7.4 shows a plot of these curves for large valued scores ($\ell > -10$).

A mapping to large posterior probabilities at low scores can be seen for {`sil`}. This indicates a large number of low target scores. The stair shape of {`Z`}, caused by a very small amount of available training data, can easily be seen in contrast to the mostly smooth curves of the other two phonemes.

The resulting classification accuracies for the individual classifiers after PAV calibration can be seen in Tables 7.10 and 7.11. No significant increase or decrease can be seen after PAV calibration for the 44 class classification, but a major decrease is shown when classifying

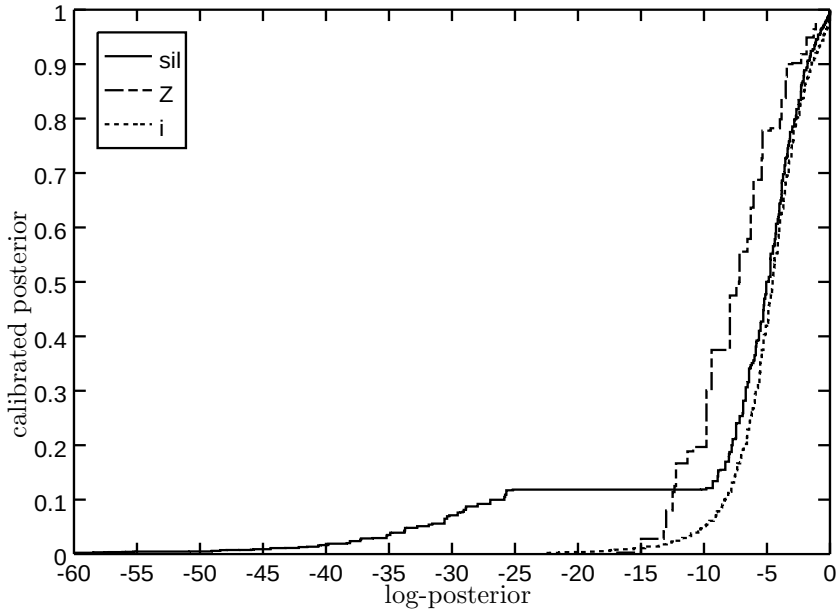


Figure 7.3: *The 44 class trained PAV curves with an equal number of target and non-target scores for the phonemes* {sil}, {Z} *and* {i}. *These phonemes represent an example of the most frequent phoneme, one of the least frequent and a typically occurring phoneme.*

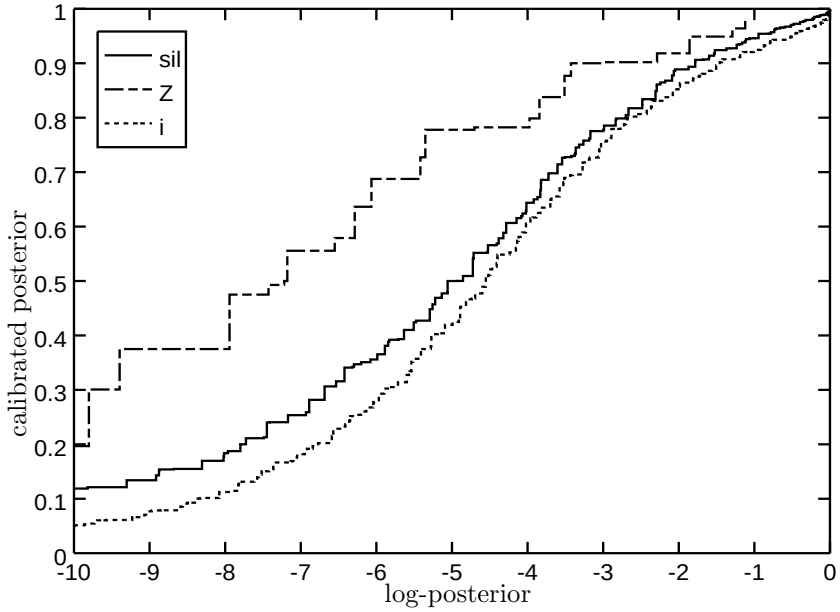


Figure 7.4: *The 44 class trained PAV curves (only plotted for* $\ell > -10$*)) with an equal number of target and non-target scores for the phonemes* {sil}, {Z} *and* {i}. *These phonemes represent an example of the most frequent phoneme, one of the least frequent and a typical occurring phoneme.*

137 classes. This decrease might be attributed to the large number of classes and the low amount of training data available for some classes.

	original	after PAV
AE	45.59%	45.04%
BE	39.78%	39.54%
CE	44.66%	44.60%
EE	45.83%	46.26%
IE	43.35%	43.69%
ALL	50.30%	50.11%

Table 7.10: *The classification accuracies for the individual classifiers (44 class) after PAV calibration with an equal number of target and non-target scores. No significant increase or decrease in classification accuracy is shown.*

	original	after PAV
AE	39.92%	28.00%
BE	33.65%	24.40%
CE	38.72%	26.12%
EE	40.49%	27.50%
IE	37.13%	26.10%
ALL	47.75%	30.06%

Table 7.11: *The classification accuracies for the individual classifiers (137 class) after PAV calibration with an equal number of target and non-target scores. A major drop in accuracy can be seen after PAV calibration. Too many classes with not enough training data for some might be the cause.*

When we look closer at the per-class accuracies, some classes show increases, some show no difference and others show decreases. The same effect can be seen in the 44 class case, but with less consequence on the total classification accuracy. Tables 7.12 and 7.13 lists the per class accuracies before and after PAV calibration for the 44 class `EE` classifier, sorted in ascending order of the frequency of occurrence. For example, these results show an increase of 21.05% for {`D`}, 12.5% for {`tS`} and 11.94% for {`sil`}. Unfortunately it also shows decreases of 42.11% for {`d\`}, 82.32% for {`r\`} and 35.71% for {`other`}. In the 44 class case these increases and decreased average to about the same accuracy, but unfortunately in the 137 case, the decreases far outnumber the increases.

After PAV calibration these classifiers are fused by a basic score averaging fusion. The results are shown in Table 7.14. Fusing the 44 class classifier after calibration shows an increase in accuracy when compared to our baseline `ALL` classifier, but when compared to the 53.98% of the uncalibrated fusion in Section 7.7, we can see that the calibration does

	phoneme	original	after PAV	% accuracy increase
1	d\	19 %	11 %	-42.11 %
2	Z	32 %	36 %	12.50 %
3	Oi	45 %	49 %	8.89 %
4	u@	8 %	8 %	0.00 %
5	iu	25 %	21 %	-16.00 %
6	g	37 %	39 %	5.41 %
7	D	38 %	46 %	21.05 %
8	h	44 %	46 %	4.55 %
9	dZ	39 %	43 %	10.26 %
10	tS	40 %	45 %	12.50 %
11	N	46 %	46 %	0.00 %
12	i@:	39 %	44 %	12.82 %
13	r\	17 %	3 %	-82.35 %
14	S	56 %	56 %	0.00 %
15	au	46 %	51 %	10.87 %
16	b	43 %	44 %	2.33 %
17	p	37 %	37 %	0.00 %
18	z	27 %	21 %	-22.22 %
19	j	42 %	45 %	7.14 %
20	r	44 %	47 %	6.82 %
21	v	39 %	39 %	0.00 %
22	T	28 %	20 %	-28.57 %

Table 7.12: *The classification accuracies for the first half of the phoneme classes (phonemes* {d\} *to* {T}*), sorted from least to most occurring, of the 44 class* ***EE*** *classifier before and after PAV calibration. An equal number of target and non-target scores were used for the PAV training.*

	phoneme	original	after PAV	% accuracy increase
23	ae	38 %	41 %	7.89 %
24	d	35 %	34 %	-2.86 %
25	m	39 %	38 %	-2.56 %
26	a	36 %	38 %	5.56 %
27	w	61 %	62 %	1.64 %
28	other	42 %	27 %	-35.71 %
29	l	40 %	32 %	-20.00 %
30	A	44 %	42 %	-4.55 %
31	k	43 %	45 %	4.65 %
32	u	30 %	24 %	-20.00 %
33	O:	49 %	51 %	4.08 %
34	E	38 %	39 %	2.63 %
35	@	33 %	30 %	-9.09 %
36	f	52 %	53 %	1.92 %
37	@u	29 %	24 %	-17.24 %
38	ai:	47 %	49 %	4.26 %
39	@i	43 %	40 %	-6.98 %
40	t	35 %	34 %	-2.86 %
41	n	38 %	38 %	0.00 %
42	s	39 %	37 %	-5.13 %
43	i	52 %	51 %	-1.92 %
44	sil	67 %	75 %	11.94 %

Table 7.13: *The classification accuracies for the second half of the phoneme classes (phonemes* {ae} *to* {sil}*), sorted from least to most occurring, of the 44 class* ***EE*** *classifier before and after PAV calibration. An equal number of target and non-target scores were used for the PAV training.*

not improve classification accuracy. As expected from the individual classifier performance after calibration, the 137 class fusion achieves a low accuracy.

	44 class	137 class
ALL	50.30%	47.75%
Fused system after PAV calibration	52.70%	34.10%

Table 7.14: *The classification accuracies for the calibrated and fused classifiers, compared to that of our baseline* `ALL` *classifier. The fused classifiers show a slight increase in accuracy compared to the* `ALL` *classifier, but less when compared to that of the uncalibrated fusion with an accuracy of 53.98%. A major decrease in classifier accuracy is shown after PAV calibration for 137 classes.*

7.8.2 Target and non-target scores in proportion of class occurrence

In this section the previous experiment is repeated, but with the number target and non-target scores not equal, with target and non-target scores selected according to the phoneme occurrence.

Figure 7.5 shows the trained PAV curves for the phonemes {`sil`} (the most frequent phoneme), {`Z`} (one of the least frequent phonemes) and {`i`} (a typically occurring phoneme). Figure 7.6 shows a plot of these curves for large valued scores ($\ell > -10$).

From these figures we can clearly see the scores for the scarce phoneme {`Z`} being mapped to much lower posterior probabilities than those of the frequent phonemes. Tables 7.15 and 7.16 show the classification results for the individual calibrated classifiers. Unlike the experiment in the previous section, a large increase in accuracy can be seen. The PAV curves incorporates a prior probability into the calibration, biasing the classifiers to the more frequent classes.

	original	after PAV
AE	45.59%	52.46%
BE	39.78%	47.26%
CE	44.66%	52.27%
EE	45.83%	53.18%
IE	43.35%	51.23%
ALL	50.30%	57.59%

Table 7.15: *The classification accuracies for the individual classifiers (44 class) after PAV calibration with target and non-target scores in proportion of the class occurrence. A major increase in the total classification accuracy is shown.*

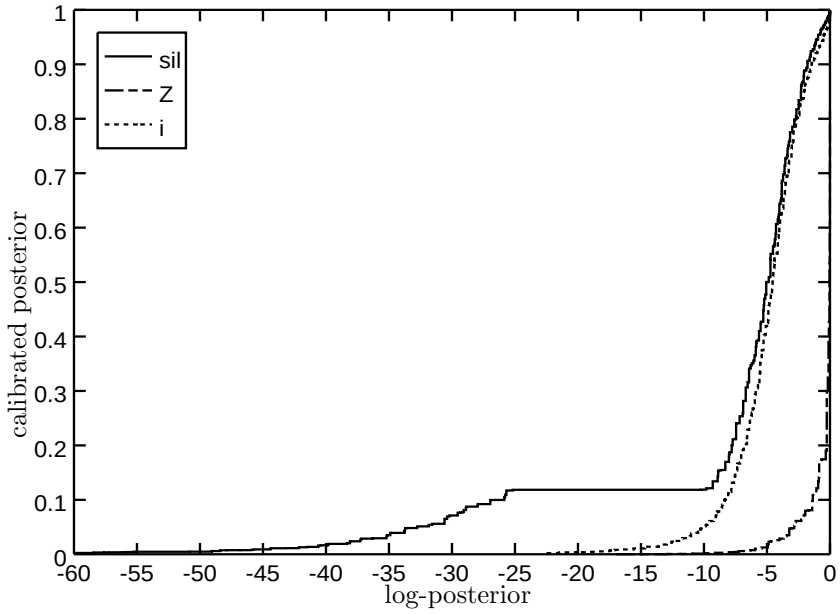


Figure 7.5: *The 44 class trained PAV curves with target and non-target scores in proportion of the class occurrence for the phonemes* {sil}*,* {Z} *and* {i}*. These phonemes represent an example of the most frequent phoneme, one of the least frequent and a typically occurring phoneme.*

	original	after PAV
AE	39.92%	42.72%
BE	33.65%	37.22%
CE	38.72%	42.07%
EE	40.49%	43.17%
IE	37.13%	40.77%
ALL	47.75%	47.42%

Table 7.16: *The classification accuracies for the individual classifiers (137 class) after PAV calibration with target and non-target scores in proportion of the class occurrence. A large increase in accuracy is shown for most classifiers.*

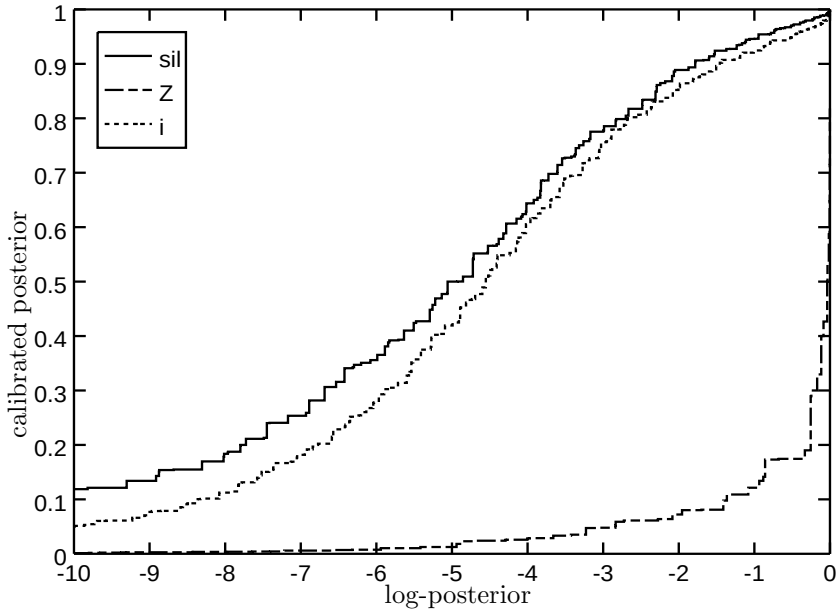


Figure 7.6: *The 44 class trained PAV curves (only plotted for $\ell > -10$)) with target and non-target scores in proportion of the class occurrence for the phonemes* {sil}, {Z} *and* {i}. *These phonemes represent an example of the most frequent phoneme, one of the least frequent and a typical occurring phoneme.*

The per class accuracies for the 44 class EE classifier can be seen in Tables 7.17 and 7.18 where the link between the frequency of each class and the resulting accuracy for that class can clearly be seen. Our choice of target and non-target scores and the resulting application of the prior by the PAV algorithm, has almost completely suppressed the least frequent classes in favour of the most frequent phonemes.

After PAV calibration these classifiers are once again fused by a basic score averaging fusion. The results are shown in Table 7.19. 137 class classification results in a slight decrease in accuracy. This might be explained by the large number of classes and the scarceness of training data for each state of each phoneme from which the classes were derived. Impressive gains in accuracy is seen when classifying 44 classes. If total system accuracy is our only metric of success, then this method achieves our goal.

7.9 C_{llr} and Linear Logistic Regression

In this section we experiment with the optimisation of the metric C_{llr} by means of Linear Logistic Regression to calibrate and fuse our five expert classifiers. Due to memory restrictions, only the 44-class case was tested.

We used the FoCal-Multi-class [4] MATLAB toolkit to train the parameters needed for the linear transformation of the output scores. The purpose of this toolkit (taken directly

	phoneme	original	after PAV	% accuracy increase
1	d\	19 %	0 %	-100.00 %
2	Z	32 %	13 %	-59.38 %
3	Oi	45 %	16 %	-64.44 %
4	u@	8 %	1 %	-87.50 %
5	iu	25 %	11 %	-56.00 %
6	g	37 %	24 %	-35.14 %
7	D	38 %	22 %	-42.11 %
8	h	44 %	31 %	-29.55 %
9	dZ	39 %	30 %	-23.08 %
10	tS	40 %	33 %	-17.50 %
11	N	46 %	23 %	-50.00 %
12	i@:	39 %	34 %	-12.82 %
13	r\	17 %	1 %	-94.12 %
14	S	56 %	55 %	-1.79 %
15	au	46 %	35 %	-23.91 %
16	b	43 %	37 %	-13.95 %
17	p	37 %	27 %	-27.03 %
18	z	27 %	14 %	-48.15 %
19	j	42 %	34 %	-19.05 %
20	r	44 %	44 %	0.00 %
21	v	39 %	34 %	-12.82 %
22	T	28 %	12 %	-57.14 %

Table 7.17: *The classification accuracies for the first half of the phoneme classes (phonemes* {d\} *to* {T}*), sorted from least to most occurring, of the 44 class* **EE** *classifier before and after PAV calibration. The proportion of target to non-target scores were chosen according to phoneme occurrence.*

	phoneme	original	after PAV	% accuracy increase
23	ae	38 %	31 %	-18.42 %
24	d	35 %	32 %	-8.57 %
25	m	39 %	34 %	-12.82 %
26	a	36 %	37 %	2.78 %
27	w	61 %	64 %	4.92 %
28	other	42 %	19 %	-54.76 %
29	l	40 %	35 %	-12.50 %
30	A	44 %	48 %	9.09 %
31	k	43 %	49 %	13.95 %
32	u	30 %	26 %	-13.33 %
33	O:	49 %	53 %	8.16 %
34	E	38 %	44 %	15.79 %
35	@	33 %	40 %	21.21 %
36	f	52 %	53 %	1.92 %
37	@u	29 %	29 %	0.00 %
38	ai:	47 %	58 %	23.40 %
39	@i	43 %	44 %	2.33 %
40	t	35 %	55 %	57.14 %
41	n	38 %	62 %	63.16 %
42	s	39 %	59 %	51.28 %
43	i	52 %	66 %	26.92 %
44	sil	67 %	82 %	22.39 %

Table 7.18: *The classification accuracies for the second half of the phoneme classes (phonemes* {`ae`} *to* {`sil`}*), sorted from least to most occurring, of the 44 class* ***`EE`*** *classifier before and after PAV calibration. The proportion of target to non-target scores were chosen according to phoneme occurrence.*

	44 class	137 class
ALL	50.30%	47.75%
Fused system after PAV calibration	59.40%	46.20%

Table 7.19: *The classification accuracies for the calibrated and fused classifiers, compared to that of our baseline* ***`ALL`*** *classifier. The PAV calibrations were trained from target and non-target scores in proportion to the class occurrence. A major increase in total system accuracy is seen when classifying 44 classes. The accuracy for the 137 class system slightly decreases, possibly due to the larger number of classes and greater scarceness of training data for each class.*

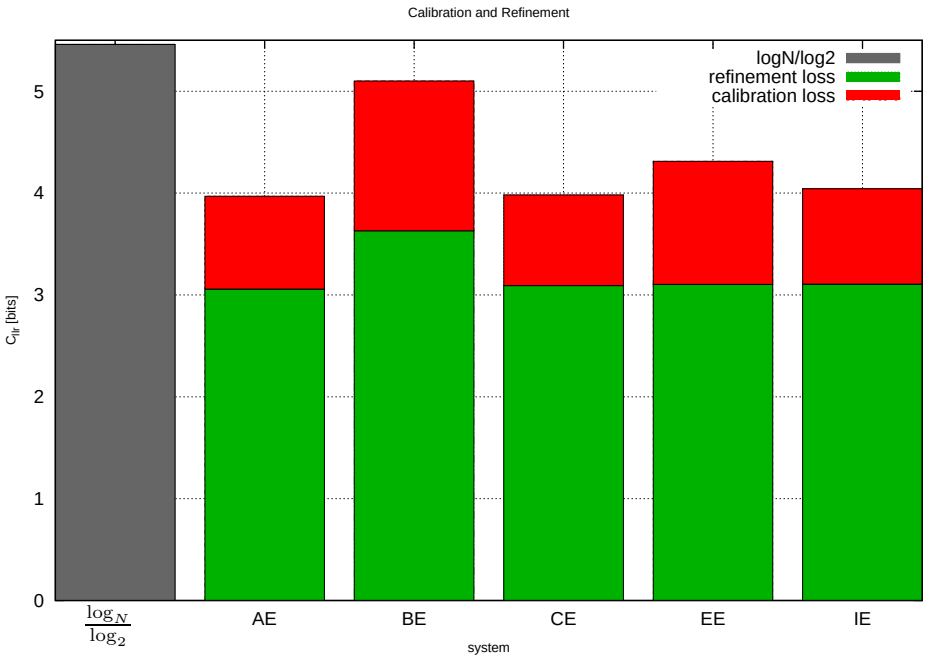


Figure 7.7: *The calibration and refinement loss of each classifier.*

from the manual) is to help the creators of classifiers to:

- Evaluate the quality of a recogniser under development.
- Fuse the outputs of multiple recognisers to form a better recogniser.
- Calibrate the recogniser output, so that it can be used to make cost-effective Bayes decisions, over a wide range of multi-class recognition applications.

The first point above refers to the calibration and refinement loss (see Chapter 5.3) indicated by C_{llr}. Figure 7.7 shows C_{llr} for each classifier and the reference $C_{llr} = \log_2 N$ for a well calibrated, but useless classifier. For each classifier we can see the calibration loss that can be recovered and the unrecoverable refinement loss. As we could clearly see in the previous results for the individual classifiers, the `BE` classifier has the highest loss, both for calibration and refinement.

We then used the toolkit to train the weights α_k and the vector β for the linear calibration and fusion transformation

$$\vec{\ell}'(x_t) = \sum_{k=1}^{K} \alpha_k \vec{\ell}_k(x_t) + \vec{\beta}, \tag{7.3}$$

with log-likelihood score vectors $\vec{\ell}_k(x_t)$ and total number of classifiers K. In this case, $K = 5$.

Table 7.20 shows the trained weight (α_k) for each classifier. The size of the weight of the `BE` classifier in relation to that of the others is very small. This is due to the large unrecoverable refinement loss. Assigning the smallest weight to the least accurate system should logically be the wise choice to make, but the results in Table 7.21 might show otherwise.

AE	0.149
BE	0.055
CE	0.124
EE	0.128
IE	0.166

Table 7.20: *The trained weight for classifiers* `AE`*,* `BE`*,* `CE`*,* `EE` *and* `IE` *(accurate to three decimal places). The very small weight trained for the* `BE` *classifier is easily noticeable.*

The results were lower than expected, with a negligible difference in accuracy between the fused expert classifiers and the `ALL` classifier. While these results were unexpected, they can be explained by closer investigation into the calculated classifier weights. The `BE` classifier might have the lowest total accuracy of all, but it is still the expert on the `BE` accent group. With the assigned weight of 0.055 (less than half of any other classifier), the opinion of this classifier is practically ignored. Ignoring the opinion of an expert while classifying data from its accent group, can be expected to reduce the accuracy for that data. The C_{llr} plot for the fused system is shown in Figure 7.8. Note the zero calibration loss, with only the unrecoverable refinement loss remaining.

original `ALL` classifier	50.30%
FoCal calibration and fusion	49.91%

Table 7.21: *The accuracies of the original* `ALL` *classifier and the FoCal calibration and fusion of the* `AE`*,* `BE`*,* `CE`*,* `EE` *and* `IE` *classifiers (44 class). There is a slight decrease in accuracy when compared to the 50.30% of our baseline ALL classifier.*

In order to avoid the low weights for weaker classifiers, we decided to FoCal calibrate each classifier individually and then fuse. The results (see Table 7.22) show some improvement that points to our theory about the `BE` classifier and its low weight in the previous experiment. An additional calibration after the fusion achieves basically the same results. C_{llr} for these two systems can be seen in Figure 7.9. The fused system from individually calibrated classifiers still shows some calibration loss that is recovered in the additional after-fusion calibration.

original `ALL` classifier	50.30%
FoCal individual calibration and fusion	51.15%
FoCal calibration after fusion	51.24%

Table 7.22: *The accuracies of the original* `ALL` *classifier, the fusion of the individually calibrated* `AE`*,* `BE`*,* `CE`*,* `EE` *and* `IE` *classifiers and the calibrated fused system (44 class). A slight increase in accuracy can be seen after calibration and fusion.*

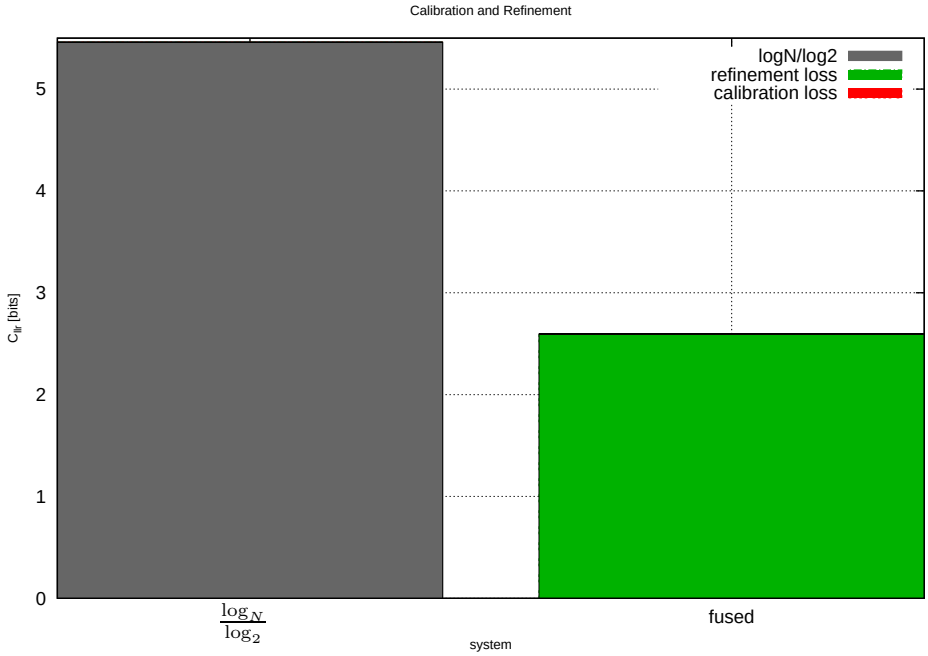


Figure 7.8: *The calibration and refinement loss of the fused system. All the calibration loss has been completely recovered.*

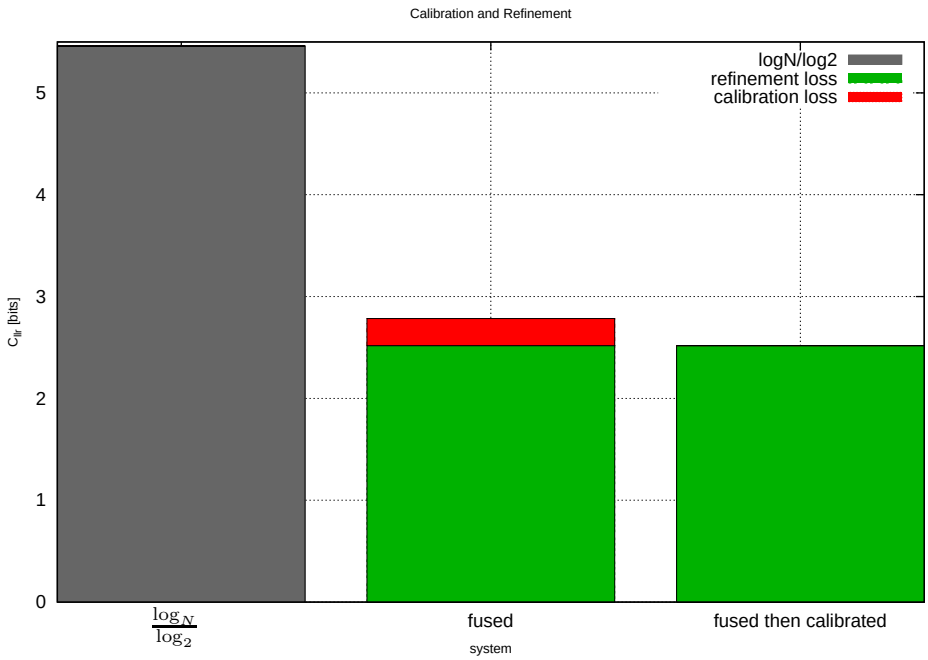


Figure 7.9: *The calibration and refinement loss of the fused system from individually calibrated classifiers, before and after an additional calibration. Some calibration loss still showed after the first individual calibration, but was recovered in the final calibration.*

	1	8	16
64 classifiers	36.72%	42.38%	43.86%
128 classifiers	36.72%	42.36%	43.85%
256 classifiers	36.72%	42.35%	43.80%

Table 7.23: *The resulting accuracies of 137 class Bagging classifiers for several internal classifier types (1, 8 and 16 component Gaussian Mixture Models) and total number of classifiers (64, 128 and 256). None of these achieve the 47.75% target accuracy of our baseline* `ALL` *classifier.*

7.10 Bagging and Boosting

In the previous section we had a group of classifiers that were trained on specific parts of a dataset, namely the different accent groups. We were able to fuse these systems for an accuracy higher than a single system trained on all the data. In this section we experiment with the Bagging and Boosting algorithms (see Chapter 6), in which various classifiers are once again fused, but the training data for each is automatically chosen (and weighted) by the algorithm.

For both Bagging and Boosting we experimented with several different classifier types. Diagonal Gaussian Models were chosen as an example of a model that fails at capturing the essence of a phoneme. GMMs (Gaussian Mixture Models) with eight Diagonal Gaussian components were the next choice. These models are still not very accurate, but at least much better than guessing. For the final experiments we chose GMMs with 16 Diagonal Gaussian components. These models should be adequate at classifying phonemes, but still much less accurate than the phoneme models used in the previous sections, and therefore might be a good choice for a *weak* classifier.

Table 7.23 shows the resulting accuracies for the nine test systems. The accuracy of the most complex models reach 43.9% in comparison to the 47.7% of the `ALL` classifier. The same situation can be seen in tests for the 44 class case, shown in Table 7.24. Another aspect to note is that higher numbers of classifiers do not increase the total accuracy. We expected the 256 classifier system to perform better than the 128 classifier system and much better than the 64 classifier system, but as shown in the results, this is not the case.

While the Bagging classifiers did not achieve the target accuracy, these results might point to the possibility that the Bagging algorithm is not feasible for classification for a large number of low accuracy classes, as in the case of phoneme recognition.

More complex models could be trained and used as weak classifiers for the Bagging algorithm, but that would defy one of the main reasons for using Bagging: classifiers should be quick and easy to train. As soon as the classifier from each iteration of the algorithm approaches training times in the order of a single "strong" classifier, training times become unpractical.

	1	8	16
64 classifiers	40.69%	46.33%	47.73%
128 classifiers	40.68%	46.30%	47.71%
256 classifiers	40.68%	46.33%	47.67%

Table 7.24: *The resulting accuracies of 44 class Bagging classifiers for several internal classifier types (1, 8 and 16 component Gaussian Mixture Models) and total number of classifiers (64, 128 and 256). None of these achieved the 50.30% target accuracy of our baseline* `ALL` *classifier.*

	1	8	16
64	36.38%	42.71%	43.00%
128	36.89%	41.68%	44.09%
256	36.78%	42.81%	43.92%

Table 7.25: *The resulting accuracies of 137 class Boosting classifiers for several internal classifier types (1, 8 and 16 component Gaussian Mixture Models) and total number of iterations (64, 128 and 256). None of these achieve the 47.75% target accuracy of our baseline* `ALL` *classifier.*

Tables 7.25 and 7.26 shows the accuracies of the Boosting classifiers. The results closely resembles those of the Bagging classifiers and the same conclusion can be made. We expected the Boosting classifiers to outperform the Bagging classifiers. While these results were not what we expected, due to time restrictions, further investigation could not be attempted.

7.11 Summary

In this chapter we introduced our test systems and the experimental results obtained with the techniques described in the previous chapters. The first section covers the AST South African English dataset that we used for training and testing. We commented on the need for a database consisting of accent group subsets, specifically for a language such as South African English with many variations in pronunciation.

The structure of our basic fusion system was outlined next. We defined the models used and the classifiers to be fused. A fusion of the five classifiers `AE`, `BE`, `CE`, `EE` and `IE` was compared to the single system trained on all accent groups, `ALL`.

The next section describes a mapping from the 137 phoneme classes to 44 classes. Our phoneme set consists of 44 phonemes in total, but for modelling purposes most were divided into several states, thus resulting in 137 classes. We applied Bayes' Theorem with a suitable prior to map likelihood scores to posterior probabilities. After this mapping, the posterior probabilities of all the states of the same phoneme can be added together to form one

	1	8	16
64	39.64%	45.02%	45.50%
128	40.14%	44.42%	46.26%
256	39.94%	45.14%	45.88%

Table 7.26: *The resulting accuracies of the 44 class Boosting classifiers for several internal classifier types (1, 8 and 16 component Gaussian Mixture Models) and total number of iterations (64, 128 and 256). None of these achieved the 50.30% target accuracy of our baseline* `ALL` *classifier.*

posterior probability for each phoneme.

We tested each individual classifier and listed the results in the next section. It was clearly shown that the system trained on all the data, `ALL`, is vastly superior in both the 137 and 44 class cases. For comparative purposes we tested the 137 class classifiers for both likelihood scores and posterior probabilities (with our chosen priors).

Our first fusion experiments then followed. Fusion from hard decisions was shown with tests covering voting systems and the phoneme characteristic system previously explained. The results were not spectacular, with only a small increase in accuracy during 44 class classification. Out of the three test systems, the basic equally-weighted voting system performed best.

The initial score fusion test system achieved basically the same accuracies as the voting systems. We expected a weighted score fusion system to achieve better results.

Calibration with the PAV algorithm was done for the flat prior (equal number of target and non-target scores) and the prior equal to the class occurrence. In the 137-class case a major decrease in accuracy was found for PAV with a flat prior. The 44-class classifiers achieves approximately the same accuracies.

Choosing the proportion of target to non-target scores according to the class frequencies resulted in impressive accuracies. The application of the prior by PAV biased the classifiers in favour of the more frequent classes and resulted in a major increase in the total classification accuracy. Impressive gains were seen with the fusion of the 44-class PAV-calibrated classifiers.

The information metric C_{llr} and the use thereof for calibration and fusion was also investigated. We showed the calibration and refinement loss for each classifier, with the `BE` classifier, as expected, suffering from the highest loss. Calibrating and fusing the five classifiers resulted in a system marginally more accurate than the target `ALL` classifier. We commented on the possible negative effect of weighing the least accurate classifier (`BE`) much lower than the other classifiers. In order to avoid this, we calibrated each classifier individually. The final fusion of these calibrated classifiers achieved more accurate results.

For Bagging and Boosting we investigated nine possible classifiers. Classifiers consisting of 64, 128 and 256 individual weak classifiers were trained. For each of these experiments, we investigated a single Diagonal Gaussian Model, an eight component GMM and finally a

16 component GMM.

None of these classifiers achieved the target accuracy of the `ALL` classifier. We attribute this to the low accuracy of our phoneme recognition system. While Bagging and Boosting is designed to work with low accuracy classifiers, the chosen classifier types might have been too inaccurate for Bagging or Boosting use. Choosing more accurate classifiers might solve this problem, but the resulting training times would be unpractical. The fact that the Boosting classifiers showed weaker performance than the Bagging classifiers was not what we expected. Unfortunately due to time constraints, these methods could not be futher investigated.

Chapter 8

Conclusion

8.1 Results

The main goal of this thesis was to investigate the feasibility of fusion for phoneme recognisers for South African English. Some techniques showed more promise than others and in some cases we were able to fuse the classifiers from each accent group (`AE`, `BE`, `CE`, `EE` and `IE`) into a single classifier than outperformed the classifier trained on all the data (`ALL`).

Out of all the techniques we expected a basic fusion system to achieve the lowest results, but a basic fusion of the five expert classifiers was able to achieve higher accuracies than the `ALL` classifier. We tried weighing each classifier with its individual accuracy, but no improvement in classification accuracy could be seen. Our implementation of a system that uses class labels as input and outputs constructed class scores did not achieve impressive results.

The basic score fusion systems performed adequately, with increases in accuracy in the lower 44 class case. Weighing each classifier with its individual accuracy did not make any difference to the classification results.

The importance of calibration before fusion was shown with the results from the Pool Adjacent Violators (PAV) experiments. While we could not increase the per class accuracy for every classifier, the total accuracy for each individual classifier and their resulting fusion was noticeably increased. Application of the prior through the PAV algorithm resulted in these impressive gains for the total system accuracy.

Fusion and calibration using the metric C_{llr} showed some promise and we were able to achieve a small increase in classification accuracy. Our best results for this technique was achieved with a system consisting of a fusion of the five expert classifiers, each individually calibrated, followed by an additional calibration. After the impressive gains showed in the experiments with the synthetic data, we expected this method to perform better. Once again we believe that the resulting accuracies might benefit from additional training data. Due to memory restrictions we were forced to test only the 44 class case, with just a small subset of the available training data used for the calibration training.

The implementation of the Bagging and Boosting algorithms showed no positive results.

We believe that these techniques are not specifically suited to that of phoneme recognition, as the *weak* classifiers trained in each iteration of these algorithms might be of too low accuracy to be of any use, even with a fusion of a multitude of these classifiers. In other classification fields a weak classifier (as referred to in Bagging or Boosting) is probably much more accurate than the typical phoneme recogniser. Training higher accuracy internal classifiers for Bagging or Boosting use would defeat one of the characteristics of these algorithms, namely that the classifier at each iteration should be quick to train. Using more accurate phoneme models for each iteration would result in unpractical training times. The results for these classifiers also invoke some suspicion, but unfortunately due to time constraints further investigation could not be attempted.

Throughout our work problems were encountered that relates to some inherent characteristics of phoneme recognisers and the speech corpora used to train them. While these are normal in the field of phoneme recognition, our attempts at fusion and calibration might have been more successful if they could be avoided. The three main issues and some of their negative effect were:

A large number of classes: Resulted in memory restrictions for some fusion and calibration techniques.

Low per-class accuracy for typical phoneme recognisers: Had a negative effect on the fusion of classifiers and the accuracy of the weak classifiers used for Bagging and Boosting.

Scarceness of data for some phonemes: Directly influenced the training and the resulting accuracies of the PAV calibration. Calibration and fusion using C_{llr} might also have been more successful with larger amounts of training data and a balanced dataset.

Our attempt at solving some of these problems was to reduce the number of classes to 44. This increased per class accuracies and in effect increased the available training data. Unfortunately the slight positive enhancement did not completely solve our problems. For some techniques the 44 classes still required a large amount of memory. Class accuracies increased, but were still fairly low. The scarce phonemes were still scarce, but the ability to use all the phoneme states as a single class resulted in slightly more available examples.

8.2 Future Work

Out of all the fusion and calibration techniques we experimented with, the PAV algorithm and C_{llr} with Linear Logistic Regression would benefit most from further study. The PAV algorithm showed impressive results and experimention with a larger, more balanced dataset could improve classification accuracies. Further investigation into applying C_{llr} should also be attempted. With the gains achieved in the experiments with synthetic data, we hope

that the same gains can be found for a phoneme recognition system if more training data could be used.

A more expansive South African English speech corpus would also supply the opportunity to further investigate the fusion of classifiers from subsets of data. We were able to fuse classifiers trained from the five accent groups to form a single more accurate classifier. Would splitting each group into a *male* and *female* subset and then training and fusing those ten classifiers increase accuracy even further?

Another possibility for further study is the fusion of the five accent groups, but with additional input from an accent recogniser. A larger weight could be assigned to the classifier from the recognised accent group. This study could also include a gender recogniser and be implemented with the system in the previous paragraph. The combination of the classified accent and gender would determine the assigned weight for each classifier during fusion.

Bibliography

[1] AYER, M., BRUNK, H., EWING, G., REID, W., and SILVERMAN, E., "An empirical distribution function for sampling with incomplete information." *Annals of Mathematical Statistics*, 1955, Vol. 26, No. 4, No. 4, pp. 641–647.

[2] BREIMAN, L., "Bagging predictors." *Machine Learning*, 1996, Vol. 24, pp. 123–140.

[3] BRÜMMER, N., "FoCal: Tools for Fusion and Calibration of automatic speaker detection systems." `http://www.dsp.sun.ac.za/~nbrummer/focal/index.htm`. 2005.

[4] BRÜMMER, N., "FoCal Multi-class: Toolkit for Evaluation, Fusion and Calibration of Multi-class Recognition Scores." `http://niko.brummer.googlepages.com/FoCal_MultiClass_Manual.pdf`. 2007.

[5] BRÜMMER, N., *Measuring, refining and calibrating speaker and language information extracted from speech*. PhD thesis, Stellenbosch University, 2007.

[6] BRÜMMER, N. and DU PREEZ, J., "Application Independent evaluation of speaker detection." *Computer Speech and Language*, April–July 2006, Vol. 20, No. 2–3, pp. 230–275.

[7] BRÜMMER, N. and DU PREEZ, J., "The PAV-Algorithm optimizes binary proper scoring rules." *unpublished paper*, 2007.

[8] BRÜMMER, N. and VAN LEEUWEN, D. A., "On calibration of language scores." *IEEE Odyssey 2006: The Speaker and Language Recognition Workshop*.

[9] CHILDERS, D. G., SKINNER, D. P., and KEMERAIT, R. C., "The Ceptrum: A Guide to Processing.." *Proceedings of the IEEE*, October 1977, Vol. 65, No. 10, pp. 1428–1443.

[10] CILLIERS, F., "Tree-based Gaussian Mixture Models for Speaker Verification." Master's thesis, Stellenbosch University, 2005.

[11] DE VILLIERS, E., "Automatic Alignment and Error Detection for Phonetic Transcriptions in the African Speec Technology Project Speech Databases." Master's thesis, Stellenbosch University, 2006.

[12] DU PREEZ, J. A., "Thesis-related discussions." 2005-2008.

[13] ERP, M. V., VUURPIJL, L., and SCHOMAKER, L., "An overview and comparison of voting methods for pattern recognition." in *Hoboken(NJ), IEEE. Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition (WFHR02*, pp. 195–200, 2002.

[14] FREUND, Y. and SCHAPIRE, R., "A decision theoretic generalization of on-line learning and an application to boosting." *Journal of Computer and System Sciences*, 1997, Vol. 55, No. 1, No. 1, pp. 119–139.

[15] GUTIERREZ-OSUNA, R., "Introduction to Pattern Recognition Lecture Notes." http://courses.cs.tamu.edu/rgutier/cs790_w02/l6.pdf. October 2008.

[16] LADEFOGED, P., *A Course in Phonetics*. Fourth edition. Thomson Learning, 2001.

[17] LAY, D. C., *Linear Algebra and its Applications*. Second edition. Addison Wesley Longman, 2000.

[18] MINKA, T., "A comparison of numerical optimizers for logistic regression." http://www.stat.cmu.edu/~minka/papers/logreg/. 2003.

[19] NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY, "Language Recognition Evaluation." http://www.nist.gov/speech/tests/lre/. October 2008.

[20] NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY, "Speaker Recognition Evaluation." http://www.nist.gov/speech/tests/lre/. October 2008.

[21] PEEBLES, P. Z., *Probability, Random Variables and Random Signal Principles*. Fourth edition. McGraw-Hill, 2001.

[22] RABINER, L. R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE*, February 1989, Vol. 77, No. 2, pp. 257–286.

[23] ROUX, J. C., "Results of the African Speech Technology (AST) Project." http://www.ast.sun.ac.za/publications/AST_Final.PDF. October 2006.

[24] VAN LEEUWEN, D. A. and BRÜMMER, N., "Channel-dependent GMM and Multi-class Logistic Regression." *Odyssey*, 2006.

[25] ZADROZNY, B. and ELKAN, C., "Transforming Classifier Scores into Accurate Multiclass Probability Estimates." *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining*, 2002.

[26] ZHI, J., ROSSET, S., ZOU, H., and HASTIE, T., "Multi-class AdaBoost." http://www-stat.stanford.edu/~hastie/Papers/samme.pdf. January 2006.

Appendix A

Phoneme Set

phoneme id	example		phoneme id	example
@	⟨a⟩go		T	⟨th⟩in
Z	⟨g⟩enre		a	h⟨u⟩t
ae	b⟨a⟩t		ai	m⟨y⟩
au	h⟨ow⟩		b	⟨b⟩at
d	⟨d⟩am		dZ	⟨j⟩ohn
d\	mu⟨dd_flap⟩y		@i	m⟨ay⟩
f	⟨f⟩un		g	⟨gh⟩olf
h	⟨h⟩it		i	cos⟨y⟩
i@	n⟨ea⟩r		iu	n⟨ew⟩
j	⟨y⟩es		k	⟨c⟩at
l	⟨l⟩ong		m	⟨m⟩an
@u	b⟨oa⟩t		n	b⟨oa⟩t
other	other		p	⟨p⟩ad
r	⟨r⟩ing		r\	⟨r⟩ing *(Afr)*
s	⟨s⟩in		sil	silence
t	⟨t⟩ea		tS	⟨ch⟩in
u	p⟨u⟩t		A	h⟨o⟩t
u@	p⟨oo⟩r		v	⟨v⟩an
w	⟨w⟩et		z	⟨z⟩ap
D	⟨th⟩is		E	p⟨e⟩t
N	si⟨ng⟩		O	s⟨aw⟩
Oi	b⟨oy⟩		S	b⟨oy⟩

Appendix B

Phoneme and Place of Articulation Groups

B.1 Phoneme Groups

b	p	g	k	t	d	d\

v	f	D	T	z	Z	s	S	h

tS	dZ

m	n	N

l	r	r\	j	w

i	u	E	@	a	O	ae	ae	A

@u	Oi	@i	ai	au	i@	u@	iu

other

sil

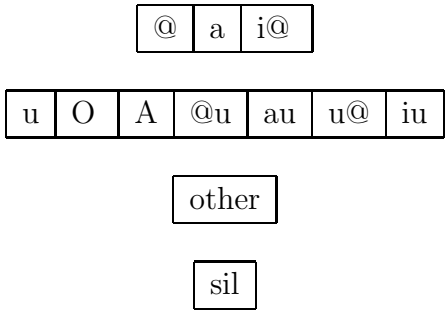
B.2 Place of Articulation Groups

b	p	v	f	m	w

t	d	d\	D	T	z	Z	s	S	tS	dZ	n	l	r	r\	j

g	k	h	N

i	E	ae	Oi	@i	ai


@
a
i@
u
O
A
@u
au
u@
iu
other
sil